Määrittelyvaiheen vaatimusten vaikutukset käyttöliittymään

Jani Hanhisalo

Pro gradu -tutkielma

Helsinki 7.12.2007

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET

Tiedekunta/Osasto	Laitos – Institution
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä – Författare
Jani Hanhisalo

Työn nimi – Arbetets titel
Määrittelyvaiheen vaatimusten vaikutukset käyttöliittymään

Oppiaine – Läroämne
Tietojenkäsittelytiede

Työn laji – Arbetets art	Aika – Datum	Sivumäärä – Sidoantal
Pro gradu -tutkielma	7.12.2007	73 + 1 liitesivu

Tiivistelmä – Referat

Tässä työssä tutkitaan, miten vaatimusmäärittelydokumenttiin kirjatut vaatimukset kiinnittävät myöhemmissä ohjelmistokehitysprosessin vaiheissa tehtäviä käyttöliittymäratkaisuja. Lisäksi tutkitaan, jääkö käyttöliittymästä puuttumaan todellisissa käyttötilanteissa tarvittavia toimintoja, kun vaatimusmäärittely toteutetaan perinteisen vesiputousmallin mukaisesti.

Tutkimuksessa tarkastellaan kahden Helsingin yliopiston ohjelmistotuotantoprojekti-kurssilla toteutetun opiskelijaprojektin tuotoksia. Tutkimuksessa selvitetään oppilaiden tuottamien käyttöliittymien keskeisimmät käyttöliittymäongelmat simulointitestaamalla käyttöliittymät. Testauksessa simuloitiin ohjelmiston kolme keskeisintä käyttötilannetta, jotka selvitettiin tekemällä kontekstuaalinen käyttäjähaastattelu yhdelle opettajatuutorille. Tämän jälkeen etsittiin, löytyvätkö ongelmien syyt ryhmien vaatimusmäärittelydokumentteihin kirjatuista käyttötapauskuvauksista tai muista vaatimuksista.

Tämän työn keskeisimpinä tuloksina selvisi, että käyttötapaukset sitoivat aina toiminnon ja sen toteutuksen käyttöliittymässä, mutta vain pieni osa niistä kiinnitti käyttöliittymäratkaisuja haitallisesti. Vakavien tehokkuusongelmien ja järjestelmästä puuttuvan toiminnallisuuden syyt kuitenkin olivat nimenomaan vaatimusmäärittelydokumentin käyttötapauksissa. Muut vaatimukset kiinnittivät toimintoja niin korkealla tasolla, ettei niistä muodostunut ongelmallisia käyttöliittymäratkaisuja. Lisäksi havaittiin, että molemmista vaatimusmäärittelyistä oli jäänyt pois sellaisia toimintoja, joita oltaisiin tarvittu käyttötilanteen suorittamiseen tehokkaasti. Vaikuttaisi siltä, että vaatimusmäärittelyvaiheessa ei ole saatu selville käyttäjän todellisia käyttötilanteita, minkä seurauksena vaatimuksista on jäänyt pois oleellisia toimintoja.

Aiheluokat (ACM Computing Classification System 1998):

D.2.1 Requirement/specifications
H.5.2 User Interface
K.6.3 Software Management: Software process

Avainsanat – Nyckelord
Käyttöliittymät, käyttöliittymäsuunnittelu, vaatimusmäärittely, ohjelmistoprosessi, vesiputousmalli

Säilytyspaikka – Förvaringställe
Kumpulan tiedekirjasto, sarjanumero C-2007-

Muita tietoja – Övriga uppgifter

Sisällys

1. Johdanto

Järjestelmän hyödyllisyys- ja käytettävyysongelmien syntyminen lähtee usein jo vaatimusmäärittelystä, jossa on kiinnitetty järjestelmän toiminnallisuutta, tietosisältöä ja käyttöliittymäratkaisuja tuntematta käyttäjien työtä kunnolla. Tämän seurauksena saatetaan jo ennen varsinaista käyttöliittymäsuunnittelua kiinnittää käyttöliittymän toiminnallisuutta haitallisesti, tai pahimmassa tapauksessa saattaa käyttöliittymästä jäädä työnkulkujen suorittamiseen vaadittavia toimintoja pois.

Helsingin yliopistolla järjestetään lukukausittain ohjelmistotuotantoprojekti-kurssi. Kurssilla oppilaiden tehtävänä on toteuttaa ohjelmistoprojekti. Projekti toteutetaan jollakin prosessimallilla. Yleisin on vesiputousmalli. Vesiputousmallissa prosessi jaetaan seuraaviin osiin: vaatimusmäärittely, suunnittelu, toteutus ja testaus.

Opiskelijaprojekteissa käyttöliittymän huomioiminen jää yleensä hyvin pieneen rooliin. Yleensä käyttöliittymä toteutetaan suunnitteluvaiheessa käyttämättä mitään systemaattista menetelmää. Viime vuosina projektiryhmät ovat alkaneet tehdä jo vaatimusmäärittelyvaiheessa ensimmäisiä prototyyppejä käyttöliittymästä. Käyttöliittymän suunnitteluun ei kuitenkaan ole mitään erityistä menetelmää, vaan jokainen projekti toteuttaa sen omia taitojansa parhaiten käyttäen. Näin toteutettuna prototyyppi ei välttämättä vastaa käyttäjän työnkulkuja, koska sen pohjana ei ole tietoa käyttäjän aidoista työnkuluista, vaan projektiryhmän ja asiakkaan esittämät vaatimukset.

Tässä työssä tutkitaan Opeapu- ja Opeapuri -opiskelijaprojekteja. Ensimmäiseksi selvitetään, miten paljon vaatimusmäärittelyvaiheessa kirjatut käyttötapaukset, käyttäjävaatimukset ja järjestelmävaatimukset sitovat käyttöliittymäratkaisuja. Lisäksi selvitetään, minkä tyyppisiä vaatimuksia ne ovat ja aiheuttavatko ne ongelmallisia käyttöliittymäratkaisuja. Toiseksi selvitetään, aiheuttaako perinteisellä tavalla tehty vaatimusmäärittely käyttöliittymään selviä tehokkuusongelmia (efficiency) ja jääkö käyttöliittymästä puuttumaan käyttäjän käyttötilanteiden kannalta tarpeellista toiminnallisuutta tai tietosisältöä, koska mukana ei ole aitoja käyttäjiä.

Luvuissa 2 ja 3 käsitellään vesiputousmalli yleisellä tasolla. Erityisesti keskitytään vaatimusmäärittelyvaiheeseen (luku 3). Luvuissa esitellään menetelmän heikkoudet ja vahvuudet käyttöliittymäsuunnittelun näkökulmasta. Seuraavassa luvussa esitellään GUIDe-prosessimalli (luku 4) ja sen tarjoamia ratkaisuja vaatimusmäärittelyvaiheeseen. Luvussa 5 esitellään tämän työn tutkimuskysymykset ja tutkitut opiskelijaprojektit.

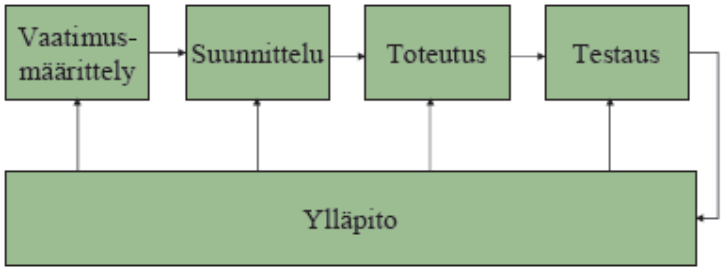
Luvussa 6 esitellään tutkimuksen toteutus ja keskeisimmät tulokset. Luvussa 7 selvitetään löytyykö ryhmien käyttöliittymistä havaittujen ongelmien syyt vaatimus-määrittelydokumenttiin kirjatuista vaatimuksista ja käyttötapauksista. Seuraavassa luvussa analysoidaan tutkimuskysymykset tutkimuksesta saatujen tulosten pohjalta ja esitetään kehitysideoita Helsingin yliopiston ohjelmistotuotantoprojektit -kurssiin (luku 8).

2. Vesiputousmalli

Vesiputousmalli on kehitetty jo vuonna 1970. Royce esitteli mallin, joka pohjautui hyvin pitkälle sen aikaisiin menetelmiin perinteisissä insinööritehtävissä [Roy70]. Sen tärkein idea oli pilkkoa ohjelmistoprosessi osiin niin, että jokaisen vaiheen lopputulosta voitiin käyttää seuraavan vaiheen syötteenä [Gus05]. Alkuperäistä versiota on paranneltu vuosien aikana, ja mallia on käytetty useiden uudempien mallien ideoiden perustana. Edelleen tänäkin päivänä sitä käytetään monien projektien prosessimallina. Neilin ja Laplanten vuonna 2002 tekemässä kyselytutkimuksessa 35 % vastanneista ilmoitti käyttävänsä vesiputousmallia edelleen prosessimallinaan [NeL03]. Tärkein syy tähän on prosessin rakenne, jonka ansiosta sen hallittavuus on selkeää [PeP00].

2.1 Vesiputousmallin rakenne

Mallin vaiheiden jaottelu vaihtelee kirjoittajan ja käyttötarkoituksen mukaan. Tässä työssä käsiteltävät projektit käyttävät hyvin pitkälle Sommervillen esittelemää mallia [Som01, s. 45]. Kuvassa 1 on kuvattu prosessin vaiheet. Tutkittavissa projekteissa ei toteuteta ylläpitovaihetta, koska opiskelijaprojektit päättyvät järjestelmän luovuttamiseen asiakkaalle.



Kuva 1. Vesiputousmallin tehtäväjako [Tai07].

Vaatimusmäärittelyvaiheessa määritellään, mitä ohjelmiston tulee tehdä. Suunnitteluvaiheessa suunnitellaan, miten tämä tullaan tekemään. Silloin määritellään tarkka kuvaus ohjelmiston komponenteista, suunnitellaan mahdolliset tietokantaratkaisut jne. Suunnittelun tuloksena syntyy suunnitteludokumentti, jonka pohjalta ohjelmisto tulee olla mahdollista toteuttaa suoraviivaisesti.

Toteutusvaiheessa ohjelmoidaan suunnitteludokumentissa kuvatut komponentit. Suunnittelu on tärkeää tehdä riittävällä huolellisuudella, jotta toteutusvaiheessa ei enää tarvitse palata suunnitteluun, vaan ohjelmisto voidaan toteuttaa tehokkaasti hyvinkin

lyhyessä ajassa. Toteutusvaiheessa tehdään yleensä myös yksikkötestaus, jossa testataan yksittäiset komponentit, kuten esimerkiksi yksittäiset luokat. Ohjelmakoodin valmistuttua tehdään testausvaiheessa laajempien kokonaisuuksien testit. Testausvaiheessa toteutetaan integraatiotestaus, jossa komponenttien välinen yhteistyö testataan. Komponenteista kootaan osajärjestelmiä ja niiden yhteistyö testataan. Lopuksi suoritetaan järjestelmätestaus, jossa koko järjestelmä testataan aidossa käyttöympäristössä.

2.2 Vesiputousmallin vahvuudet ja heikkoudet

Mallin hyvät ja huonot puolet ovat seurausta sen lineaarisesta rakenteesta. Vesiputousmalli on helppo omaksua, koska sen jokaisella vaiheella on selkeä tehtävä, ja on selvää, mihin mikäkin vaihe päättyy. Samasta syystä myös kustannusarvioiden ja aikataulujen tekeminen on helpompaa kuin prosessimalleissa, joissa on useita syklejä. Mallin pohjalta syntyy myös hyvä dokumentaatio ohjelmistosta.

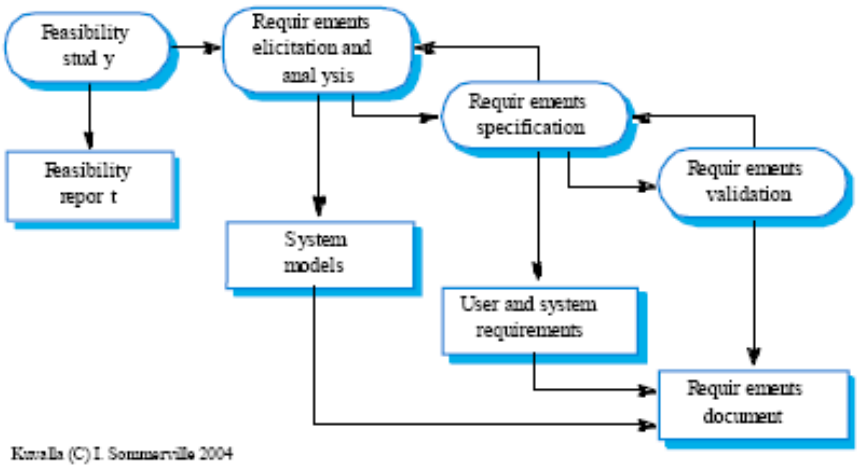
Valitettavasti oikeat ohjelmistoprojektit ovat harvoin lineaarisia [Pre97, s.35]. Suurin syy tähän on se, että on hyvin vaikeaa löytää ja varmistaa vaatimusmäärittelyvaiheessa kaikkien vaatimusten oikeellisuus [Som01, luku 6]. Lisäksi vaatimukset voivat muuttua prosessin myöhemmissä vaiheissa, esimerkiksi mikäli sidosryhmät eivät ole täysin ymmärtäneet, mitä vaatimusmäärittelyvaiheessa on sovittu [Rob03]. Vaatimuksista on myös voinut jäädä pois toimintoja, jotka ovat käyttäjien työnkulkujen kannalta välttämättömiä, jolloin ne on lisättävä vaatimuksiin. Myöhemmissä vaiheissa voidaan myös havaita, että vaatimukset ovat ristiriidassa toistensa kanssa. Tällöin joudutaan muokkaamaan vaatimusmäärittelydokumenttiin kirjattuja vaatimuksia, mikä on vesiputousmallissa vaikeaa. Vesiputousmalli on myös kallis prosessimalli, koska menetelmä perustuu dokumentoinnin kautta etenemiseen. Dokumenttien laatiminen vie aikaa, ja muutokset dokumentteihin ovat työläitä toteuttaa [Boe06]. Käyttöliittymän kannalta prosessimallin suurin ongelma on se, että käyttöliittymän suunnittelu ei ole osana vaatimusmäärittelyä, vaan se toteutetaan vasta suunnitteluvaiheessa, jolloin on jo tehty päätöksiä ohjelmiston toiminnoista ja rakenteesta [PaK03].

3. Vaatimusmäärittely

Vaatimusmäärittelyvaiheessa määritellään ohjelmistolle toiminnot, jotka sen tulee toteuttaa. Käytännössä sovitaan ohjelmiston tilanneen asiakkaan kanssa, mitä toimintoja tulevaan ohjelmistoon toteutetaan [Som01, luku 6]. Vaatimusmäärittely on projektin tärkein vaihe, sillä siinä tapahtuvat virheet ovat kaikkein vakavimpia, ja mitä myöhemmin ne löydetään, sitä kalliimmaksi ne tulevat [EnR03, s.17, PaE03]. Tämän vuoksi on tärkeää, että vaatimusmäärittely tehdään huolella ja siihen käytetään riittävästi aikaa, jotta löydetään mahdollisimman kattavasti ohjelmiston tulevat toiminnot ja tietosisältö. Luvussa 3.1 esitellään perinteisen vesiputousmallin mukaisen vaatimusmäärittelyn vaiheet, ja luvussa 3.2 kerrotaan menetelmän vahvuuksista ja heikkouksista.

3.1 Vaatimusmäärittelyn vaiheet

Vaatimusmäärittelyvaiheen voidaan ajatella olevan itsessään prosessi, jossa on useita vaiheita. Yleisesti voidaan sanoa, että seuraavat vaiheet esiintyvät vesiputousmallin vaatimusmäärittelyssä tavalla tai toisella. Näitä ovat kartutus, analysointi ja validointi [Som01, s. 122]. Vaatimusmäärittelyyn voidaan lisäksi liittää ”human to machine interface” -suunnittelu (HMI) [Bra02, luku 2.4]. Käytännössä tämä tarkoittaa alustavaa käyttöliittymäsuunnittelua. Sommervillen mallissa vaatimusmäärittelyn alussa on käyttökelpoisuustutkimus (feasibility study), jossa arvioidaan, onko ohjelmistoa järkevää lähteä toteuttamaan [Som01, s. 122-123]. Kuvassa 2 on Sommervillen jaottelu.



Kuva 2. Vaatimusmäärittelyprosessin vaiheet [Som01, s.122].

Prosessin ensimmäinen vaihe on kartutus. Kartutuksessa kerätään eri sidosryhmiltä tietoa ohjelmiston käyttöympäristöstä ja heidän vaatimuksistaan ohjelmistolle [Pae03]. Tuloksena on joukko muistiinpanoja ja ymmärrystä toimintaympäristöstä [Bray02, luku 2.2]. Vaatimusten selvitys aloitetaan yleensä haastattelemalla asiakasta. Vaatimusten esiin saaminen voi olla työlästä, koska asiakas ei aina tiedä, mitä haluaa, tai hän ei osaa selittää sitä niin, että vaatimusmäärittelyä tekevä henkilö ymmärtäisi täysin, mitä hän haluaa [Lau02, s. 4]. Vaatimusten kartutuksessa voidaan haastattelujen lisäksi käyttää skenaarioita, käyttötapauksia, järjestelmän käyttäjien tarkkailua, ryhmäkeskusteluja, aivoriihiä ja monenlaisia prototyyppejä [Pae03].

Analyysivaiheessa pyritään kartutuksen tietojen pohjalta rakentamaan kuvaus järjestelmästä. Tässä vaiheessa hahmotellaan ohjelmiston rakenne ja pyritään mahdollisimman selvästi kuvaamaan, mitä valmiin ohjelmiston tulisi tehdä. Analyysin aikana löydetyt vaatimukset kirjataan vaatimusmäärittelydokumenttiin (requirements specification). Dokumenttiin kirjataan käyttäjävaatimukset, jotka kuvaavat ohjelmiston toiminnot ja sille asetettavat rajoitukset. Käyttäjävaatimuksia käytetään yhteydenpidossa asiakkaan kanssa. Onkin tärkeää, että niiden kuvaukset on tehty sellaisella kielellä, että asiakas ymmärtää niiden sisällön. Käyttäjävaatimukset kirjataan vaatimusmäärittely-dokumenttiin luonnollisella kielellä [Som01, luku 5]. Vaatimusmäärittelydokumenttiin kuvataan myös järjestelmävaatimukset. Ne ovat hyvin lähellä käyttäjävaatimuksia, mutta niissä kuvataan järjestelmän tarjoamat palvelut ja järjestelmälle asetettavat rajoitukset yksityiskohtaisemmin. Järjestelmävaatimukset toimivat usein sopimuksena asiakkaan ja yrityksen välillä [Som01, luku 5]. Lisäksi vaatimusmäärittelydokumentissa määritellään toimintaympäristövaatimukset. Niillä kuvataan ohjelmiston ympäristön asettamat vaatimukset ohjelmistolle. Tällaisia voivat olla esimerkiksi vaatimus tietyn tietokannan käytöstä [Pfl98].

Vaatimukset voidaan jakaa toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. Toiminnalliset vaatimukset kuvaavat järjestelmän tarjoamat palvelut ja sen, miten järjestelmä toimii määritellyissä tilanteissa [Som01, luku 5]. Ei-toiminnalliset vaatimukset taas asettavat järjestelmälle rajoituksia ja reunaehtoja [Som01, luku 5]. Ei-toiminnalliset vaatimukset koskevat koko järjestelmää, joten kaikkien toiminnallisten vaatimusten on toteutettava ne. Ei-toiminnalliset vaatimukset voivat liittyä esimerkiksi ohjelmiston vasteaikoihin, luotettavuuteen tai siirrettävyyteen. Tällainen voisi olla esimerkiksi vaatimus, että ohjelmiston on toimittava Oracle- ja mySQL-tietokantojen kanssa.

Viimeiseksi vaatimusten validointivaiheessa pyritään varmistamaan, että tuotetut vaatimukset vastaavat asiakkaan toiveita ohjelmistosta ja että ohjelmistossa on varmasti

kaikki tarpeelliset toiminnot [Som01, luku 3.5]. Validoinnissa voidaan käyttää useita erilaisia menetelmiä, kuten tarkastukset, katselmointi, prototyypit tai testitapausten luonti [Som01, luku 6.3]. Helsingin yliopiston tietojenkäsittelytieteen laitoksen opiskelija-projekteissa validointi tapahtuu katselmointien avulla. Katselmoinnissa jokainen osallistuja lukee osan dokumentista ennen varsinaista katselmointia ja kirjaa muistiin kaikki havaitsemansa puutteet ja virheet. Lukemisen avuksi annetaan yleensä lista yleisimmistä ongelmista. Katselmoinnissa kirjataan muistiin havaitut ongelmat. Katselmoinnin lopuksi päätetään, voidaanko dokumentti hyväksyä sellaisenaan, hyväksyä muutoksin tai mahdollisesti hylätä, jolloin järjestetään uusi katselmointi korjausten jälkeen [Som01, luku 6.3].

3.2 Perinteisen vaatimusmäärittelyn heikkoudet

Perinteisellä vaatimusmäärittelyllä tarkoitetaan tässä luvussa 3.1 kuvattua prosessia. Siinä pyritään luonnollisella kielellä määrittelemään, miten ohjelmiston tulisi toimia. Vaiheeseen on kehitetty vuosien varrella muitakin kuvaustapoja kuin pelkkä teksti. Tunnetuimpia lienevät erilaiset kaaviot, kuten luokkakaaviot, tilakaaviot, tietovuokaaviot ja käyttötapauskaaviot. Kaavioista voidaan päätellä jonkin verran toimintojen rakenteesta, mutta silti ne kuvaavat usein enemmän ohjelmiston toimintaa kuin sitä ongelmaa, joka käyttäjän pitäisi ratkaista. Näin ollen niiden avulla ei voida varmistaa, että ohjelmisto toimisi niin, että käyttäjän on mahdollista suorittaa ohjelmistolla työntehtäviensä vaatimat toiminnot. Kaaviot siis keskittyvät enemmän yksittäisten toimintojen suorittamiseen kuin laajempien toimintoketjujen suunnitteluun.

Kaaviot parantavat monimutkaisten asioiden hahmottamista, mutta pelkkä teksti ja kaaviot eivät kuitenkaan mahdollista ohjelmiston testaamista. Siten vaatimusmäärittely-vaiheessa on hyvin vaikeaa varmistaa, tukeeko määritelty ohjelmisto käyttäjän työnkulkuja. Ohjelmiston hyödyllisyyden ja käytettävyyden testaamiseen päästään vasta ensimmäisten käyttöliittymäversioiden valmistuessa suunnitteluvaiheessa. Pahimmassa tapauksessa testaus aloitetaan vasta toteutusvaiheessa ohjelmiston ensimmäisten versioiden valmistuttua. Virheiden korjaaminen näin myöhäisessä vaiheessa on hyvin työlästä ja kallista [Som01, s. 46].

Perinteisessä vaatimusmäärittelyssä vaatimusten kartutus tapahtuu yleensä haastattelemalla asiakasta. Usein on kuitenkin niin, että asiakas ei ole järjestelmän lopullinen käyttäjä vaan mahdollisesti tilaavan yrityksen johtoon kuuluva henkilö. On hyvin mahdollista, että asiakkaan ymmärrys käyttäjän työnkuluista ei ole täysin oikea, jolloin nämä virheet tulevat myös ohjelmiston määrittäviin vaatimuksiin.

Mikäli käyttäjiä on ymmärretty haastatella, haastattelussa ei ole välttämättä varsinaisesti käytetty mitään metodia, vaan käyttäjiltä on vain kysytty, miten he nykyään tekevät työnsä ja mitä he ohjelmistolta haluavat. Näin käyttäjän työnkuluista jää helposti pois oleellisia tietoja tai niihin tulee väärää tietoa. Käyttäjän on vaikea kertoa tarkasti, miten hän työnsä tekee, koska suurin osa työssä tehtävistä toiminnoista on niin rutiinia, että käyttäjät eivät pidä niiden kertomista tärkeänä [BeH99, s. 35-37]. On myös havaittu, että kun käyttäjät kertovat, miten he työn tekevät, he eivät kuitenkaan oikeasti tee työtä tavalla, jonka he kertovat [WiR96, s. 308]. Lisäksi haastatteluja ei yleensä toteuteta käyttäjän työskentelypisteessä. Tästä syystä haastateltavalla ei välttämättä ole mukana kaikkia apuvälineitä, joita hän työskentelypisteessä käyttäisi, kuten esimerkiksi kalenteri tai erityyppiset muistilaput. Mikäli näin tehtäisiin, voitaisiin helpommin ymmärtää nykyisiä työskentelytapoja [Wix02].

Perinteisessä vaatimusmäärittelyssä pyritään kyllä selvittämään, mitä ohjelmiston tulisi tehdä. Menetelmiä ei ole kuitenkaan määritelty riittävän tarkasti. Tällöin selvitystä tekevän henkilön henkilökohtaiset taidot löytää oikeat henkilöt ja saada heiltä riittävästä informaatiota työnkuluista vaikuttavat merkittävästi oikeiden työnkulkujen löytymiseen.

Vaatimusmäärittelyn suurimmat ongelmat ovat siis käyttäjän nykyisten työnkulkujen ja niiden takana olevien tavoitteiden ymmärtäminen ja sitä kautta oikeiden vaatimusten löytymisen mahdollistaminen. Toiseksi olisi tärkeää löytää keinot vaatimusten kirjaamisen siten, että ohjelmiston toimivuutta olisi jo vaatimusmäärittelyvaiheessa mahdollista testata aidoissa käyttötilanteissa. Näin voitaisiin vähentää vesiputousmallin vaatimusten muutostarvetta ja sitä kautta parantaa vesiputousmallin toimivuutta.

4. GUIDe-prosessimallista ratkaisuja vaatimusmäärittelyyn

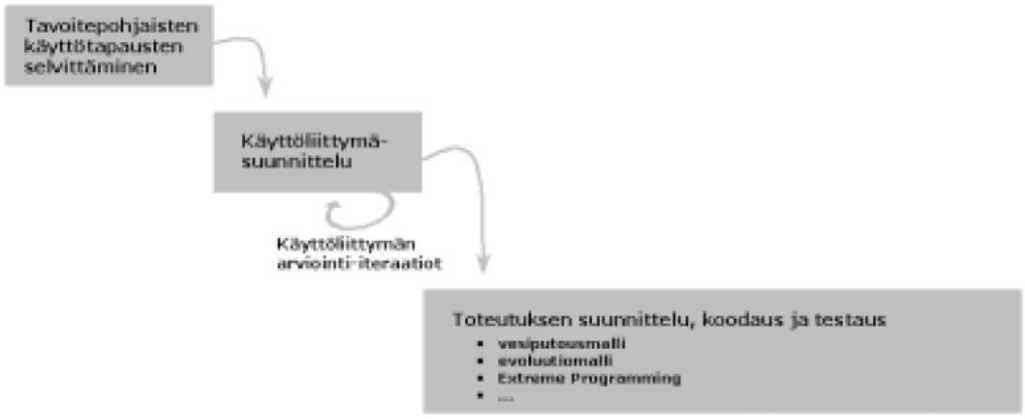
GUIDe-prosessimalli [Laa04] pyrkii käyttöliittymäsuunnittelun avulla varmistamaan jo vaatimusmäärittelyvaiheessa ohjelmiston vaatimusten oikeellisuuden. GUIDe-prosessimallin kantava idea on tuoda käyttöliittymäsuunnittelu aivan prosessin alkuun, kun perinteisissä vesiputousmallissa käyttöliittymä syntyy vasta suunnitteluvaiheessa yleensä ilman minkäänlaista käyttöliittymäsuunnitteluprosessia. GUIDessa käyttöliittymäsuunnitelma toimii vaatimuksena toteutettavalla ohjelmistolle.

Tässä työssä käsitellään vain GUIDessa käytettäviä menetelmiä, mutta kirjallisuudesta löytyy myös muita samansuuntaisia menetelmiä kuten Virtual Windows [Lau05, luku 6] ja Contextual Design [BeH98]. Näille menetelmille on yhteistä käyttäjän työtehtävien selvittäminen ja käyttöliittymän suunnittelu niiden pohjalta. GUIDen valinnan syy on se, että menetelmä on kehitetty Helsingin yliopiston tietojenkäsittelytieteen laitoksella. Lisäksi se on laitoksen käyttöliittymäsuunnittelukursseilla opetettu menetelmä, ja siihen liittyvää tutkimusta on tehty melko vähän.

Luvussa 4.1 esitellään tarkemmin GUIDe-prosessimallin vaiheet. Luvussa 4.2 käsitellään GUIDen vahvuudet ja heikkoudet. Luvuissa 4.3 - 4.5 kuvataan vaiheiden sisällä käytettäviä menetelmiä ja pyritään kuvaamaan, miten ne voisivat parantaa perinteistä vesiputousmallin vaatimusmäärittelyä.

4.1 GUIDe-prosessimallin vaiheet

Prosessimallissa on kolme keskeistä vaihetta (kuva 3): Tavoitepohjaisten käyttötapausten selvittäminen, käyttöliittymäsuunnittelu ja käyttöliittymän arviointi. Ensimmäisessä vaiheessa selvitetään kenttätutkimuksen avulla käyttäjien todelliset työnkulut ja tavoitteet. Niiden pohjalta luodaan tavoitepohjaiset käyttötapaukset [Laa04].



Kuva 3. GUIDe-prosessimallin rakenne [Laa04].

Tavoitepohjaiset käyttötapaukset ovat vesiputousmallissa määriteltyjä käyttötapauksia (use case) korkeammalla tasolla. Tavoitepohjainen käyttötapaus ei kiinnitä vielä järjestelmän toimintaa, vaan on juuri järjestelmän ulkopuolella. Se ei myöskään ole skenaario, koska skenaario kuvaa yleensä enemmän sitä miten, käyttäjä voisi suorittaa toimintoja järjestelmällä. Esimerkiksi hotellin varausjärjestelmässä käyttötapaus voisi olla huoneen varaaminen. Vastaava tavoitepohjainen käyttötapaus voisi olla esimerkiksi viikonloppuloma Kuopiossa. Tavoitepohjaisessa käyttötapauksessa kuvataan käyttäjän ongelma, tavoite ja tilannetiedot [Laa06b]. Kuvassa 4 on käyttöliittymät II -kurssin kalvoissa esitetty esimerkki tavoitepohjaisesta käyttötapauksesta. GUIDe-menetelmässä käytetään tavoitepohjaisia käyttötapauksia skenaarioiden sijaan, koska skenaarioiden avulla on vaikea päästä irti nykyisistä työskentelytavoista.

Kuva 4. Esimerkki tavoitepohjaisesta käyttötapauksesta [Laa06b].

Toisessa vaiheessa suunnitellaan käyttöliittymä tavoitepohjaisten käyttötapausten pohjalta. Menetelmänä käytetään simulointipohjaista suunnitteluprosessia [Laa06b]. Suunnitteluprosessissa käyttöliittymä suunnitellaan yksi tavoitepohjainen käyttötapaus kerrallaan, kunnes kaikki keskeisimmät käyttötilanteet on käyty läpi. Keskeisin idea on rakentaa käyttöliittymä pala palalta lisäämättä siihen mitään turhaa. Tuloksena saadaan näyttökuvasarjat siitä, miten käyttäjä suorittaa käyttötapauksissa kuvatut käyttötilanteet.

Viimeisessä vaiheessa testataan käyttöliittymäratkaisujen toimivuus. Arviointi toteutetaan käyttäen simulointitestausta [Laa06a] ja läpikäyntipalavereita (walkthroughs) [Bia91]. Havaitut virheet korjataan ja tehdään uusi arviointi. Tavoitteena on suunnitella kerralla riittävän hyvä versio, jotta iteroinnin tarve olisi vähäistä.

4.2 Mallin vahvuudet ja heikkoudet

GUIDe-prosessimallin paras puoli on se, että se mahdollistaa jo ohjelmistoprosessin alkuvaiheessa käyttöliittymäratkaisujen sekä toiminnallisten vaatimusten ja tietosisältövaatimusten testaamisen [Laa04]. Näin vältytään kalliilta muutoksilta prosessin myöhemmissä vaiheissa. Toinen tärkeä etu on, että toteutettu käyttöliittymä ja sitä kautta koko ohjelmisto perustuu käyttäjien oikeisiin työnkulkuihin eikä esimerkiksi asiakkaan antamiin kuvauksiin siitä, mitä ohjelmiston pitäisi tehdä. Tällöin ohjelmistoon tulee käyttäjän työn kannalta oleelliset toiminnot ja turhat toiminnot jäävät pois. Lisäksi käyttöliittymän kuvasarjat toimivat havainnollisena vaatimuksena ohjelmistolle ja asiakkaan on helpompi ymmärtää, millaista ohjelmistoa ollaan tekemässä.

GUIDe-prosessimallin käyttö vaatii projektin osallistujilta osaamista kenttätutkimus-menetelmistä, käyttöliittymäsuunnittelusta ja käyttöliittymän arvioinnista. Osaaminen ei kuitenkaan ole välttämätöntä, sillä GUIDen käyttö ilman aikaisempaa kokemusta saattaa olla tehokkaampaa kuin perinteinen vaatimusmäärittely, koska se mahdollistaa jo prosessin alkuvaiheessa konkreettisen käyttöliittymäversion näyttämisen asiakkaalle sen sijaan että vaatimusten hyväksyminen tapahtuu tekstimuotoisten vaatimusten perusteella. Lisäksi menetelmä kannustaa menemään käyttäjien luo, mikä jo sinänsä on edistystä pelkkään asiakkaan haastattelemiseen verrattuna. Menetelmä ei välttämättä paljasta kaikkia harvoin esiintyviä käyttötilanteita mutta toisaalta eivät perinteisetkään menetelmät sitä tee.

Keskeisin ongelma Laakson mukaan on käyttöliittymäkehityksen sijoittuminen koko projektin kriittiselle polulle, koska sen johdosta on esimerkiksi ohjelmoijat voivat joutua odottamaan käyttöliittymäsuunnittelun valmistumista [Laa04]. Toinen keskeinen riski

GUIDessa on, että se ei välttämättä paljasta hyvin harvoin toteutuvia käyttötilanteita, jolloin käyttöliittymä ei välttämättä tue niiden suorittamista, sama ongelma tosin esiintyy myös perinteisessä vaatimusmäärittelyssä. Kolmas haaste on näyttökuvasarjojen teon työläys. Mikäli järjestelmä on laaja, kuvasarjojen teko voi olla hyvinkin aikaa vievää.

4.3 Kenttätutkimusmenetelmät

Käyttöliittymäsuunnittelussa on aina ollut tärkeää ymmärtää, miten ohjelmiston käyttäjä tekee työtään ja miten ohjelmiston käyttöliittymä saataisiin tukemaan mahdollisimman hyvin käyttäjän työnkulkuja. Ilman ymmärrystä käyttäjän työnkuluista on hyvin vaikea suunnitella käyttöliittymää, jolla kyetään suorittamaan monimutkaisia, useita eri vaiheita sisältäviä työnkulkuja. Helposti päädytään käyttöliittymäratkaisuihin, jotka kyllä tukevat yksittäisiä toimintoja, mutta kokonaisten toimintoketjujen suorittaminen voi olla hyvin vaivalloista tai jopa mahdotonta.

Voidaan sanoa, että käyttöliittymäsuunnittelija tarvitsee suunnittelun pohjaksi samantyyppistä tietoa käyttäjän työtehtävistä kuin vaatimusmäärittelyn alkuvaiheissa tarvitaan. Molemmissa tarvitaan hyvin kattava käsitys siitä, miten käyttäjä tällä hetkellä suorittaa työtehtävänsä. Sen kautta on mahdollista ymmärtää, mitä toimintoja ohjelmiston tulee tarjota. Lisäksi on ymmärrettävä, mitä tietoja käyttäjä tarvitsee toimintoketjujen suoritukseen. On myös tärkeää hahmottaa, missä järjestyksessä toimintoja suoritetaan ja erityisesti miksi asiat tehdään tällä hetkellä nykyisellä tavalla.

Ymmärrys käyttäjän työtehtävistä ja tavoitteista luodaan GUIDe-prosessimallissa tekemällä kenttätutkimusta. Vaiheessa käytetään kontekstuaalisia käyttäjähaastatteluja (contextual interviews) ja käyttäjätarkkailuja (user observations) [Laa04]. Seuraavissa aliluvuissa esitellään tarkemmin, miten menetelmiä tulisi käyttää, ja eritellään niiden etuja ja haittoja.

4.3.1 Käyttäjätarkkailu

Käyttäjätarkkailussa seurataan käyttäjää, kun hän tekee työtehtäviään. Seuraamalla käyttäjän työskentelyä päästään käsiksi todellisen työtehtävän tekemisessä tarvittaviin tietoihin [BeH98, s. 47]. Tarkkailija kirjaa tarkkailun aikana muistiin mahdollisimman paljon näkemästään. Tarkkailun aikana kannattaa kerätä mahdollisimman paljon käytettyjen materiaalien (asiakirjat, lomakkeet jne.) kopioita, sillä niiden avulla on hyvä hahmottaa, mitä tietoa työnkulkuun liittyy [HaR98, s. 138-139]. Tarkkailijan on tärkeää yrittää ymmärtää käyttäjän toimenpiteiden takana olevat tavoitteet, sillä vaikka

toteutustavat muuttuisivat, tavoitteet pysyvät usein samoina [HaR98, s. 250–251]. Hyvä tapa tallentaa työnkulun vaiheita on myös digitaalisten valokuvien otto [Laa06a].

Menetelmä tuottaa luotettavaa tietoa käyttäjän työnkuluista. Tosin se voi viedä enemmän aikaa verrattuna tekniikoihin, joissa haastatellaan käyttäjää tarkkailun sijasta, mikäli tarkkailtava työvaihe on pitkäkestoinen tai harvoin tapahtuva. Tällöin parempi vaihtoehto voi olla kontekstuaalinen haastattelu.

4.3.2 Kontekstuaalinen käyttäjähaastattelu

Kontekstuaalinen käyttäjähaastattelu (contextual interview) [HaR98] on haastattelu-menetelmä, jossa käyttäjää haastatellaan hänen omassa työskentely-ympäristössään. Perusidea on antaa käyttäjälle mahdollisuus näyttää esimerkkien avulla, miten hän suorittaa keskeisimpiä työtilanteita.

Menetelmän toteuttaminen ei siis vaadi tilanteiden tapahtumista haastatteluhetkellä, vaan sen avulla on mahdollista hyvinkin lyhyessä ajassa selvittää suuri määrä käyttäjän tavoitteita ja työtehtäviä [Laa06a]. Menetelmässä haastattelijan on hyvä pyytää käyttäjää demoamaan toimintoketjujen suorittamista. Haastattelijan on siis kysymyksillään saatava käyttäjä näyttämään, miten hän esimerkiksi viimeksi suoritti työtehtävän, ja edelleen jatkokysymyksillä johdateltava käyttäjää kertomaan, miksi hän tekee asiat valitsemallaan tavalla. Haastattelijan on myös tärkeää pyrkiä löytämään työtehtävien taustalla olevat korkean tason tavoitteet, koska niiden hahmottaminen auttaa koko työprosessin hahmottamisessa ja sitä kautta mahdollistaa myös työprosessin kehittämisen.

Menetelmä vaatii haastattelijalta enemmän taitoja kyetä viemään haastattelua eteenpäin kuin käyttäjätarkkailussa vaaditaan, koska tarkkailun sijaan käyttäjä on kysymysten avulla saatava kertomaan ja demoamaan työnkulkuja. Menetelmän suurin etu on, että sen avulla on mahdollista saada hyvinkin nopeasti ohjelmiston tärkeimmät käyttötilanteet selville. GUIDe-mallissa tämä onkin olennaista, koska käyttöliittymäsuunnittelu pohjautuu kenttätutkimuksen pohjalta luotaviin tavoitepohjaisiin käyttötapauksiin.

4.4 Simulointipohjainen käyttöliittymäsuunnittelu

Vesiputousmallin kannalta on erityisen tärkeää, että vaatimusmäärittelyvaiheessa voidaan validoida vaatimukset [Lau05, s. 466]. Käyttöliittymäsuunnittelun tekeminen varhaisessa vaiheessa tarjoaa käyttäjälle ja asiakkaalle havainnollisen kuvan siitä,

minkälaista järjestelmää ollaan suunnittelemassa [Laa04]. Käyttöliittymäsuunnittelun tuloksena saatavat kuvasarjat tarjoavat välineen varmistaa projektin lopussa, että ohjelmisto toteuttaa sille asetetut vaatimukset. Perinteisessä vaatimusmäärittelyssä nämä toimenpiteet on pyritty tekemään vaatimusmäärittelydokumenttiin kirjattujen vaatimusten pohjalta. Toimintojen kuvasarjat tarjoavat kuitenkin havainnollisemman kuvan järjestelmästä ja sitä kautta parantavat vaatimusten ymmärrettävyyttä.

GUIDessa käyttöliittymäsuunnittelu toteutetaan käyttäen simulointipohjaista GDD-suunnitteluprosessia [Laa06d]. Ensimmäisen vaiheen tuloksena saadut tavoitepohjaiset käyttötapaukset toimivat käyttöliittymäsuunnittelun pohjana. Menetelmän kantava idea on suunnitella käyttöliittymä käyttötapaus kerrallaan, lisäämättä siihen mitään sellaista toiminnallisuutta tai tietosisältöä, mitä käyttötapaus ei vaadi.

Ensimmäiseksi siis valitaan yksi keskeisimmistä käyttötapauksista ja suunnitellaan mahdollisimman suoraviivainen käyttöliittymä sen vaatimien toimintojen suorittamiselle. Tämän jälkeen valitaan seuraava käyttötapaus ja lisätään käyttöliittymään sen suorittamiseen vaadittavat toiminnot. Uuden tapauksen lisäämisen jälkeen tarkistetaan, että uuden lisäys ei ole vaikeuttanut aikaisemmin lisättyjä käyttötapauksia, simuloimalla aikaisemmin lisätyt käyttötapaukset. Näin jatketaan, kunnes kaikki käyttötapaukset on käyty läpi. Mikäli kesken käyttötapauksen suunnittelun tulee mieleen toimintoja, jotka eivät liity varsinaisesti käyttötapaukseen, mutta vaikuttavat tarpeellisilta, ne kirjataan muistiin. Suunnitteluvaiheen jälkeen voidaan selvittää, löytyykö jokin tavoitepohjainen käyttötapaus, jossa suunnittelun aikana mieleen tullutta toimintoa tarvitaan. Tällaisen löytyessä se lisätään käyttötapausten listaan. Suunnittelun päätteeksi voidaan piirtää keskeisten tavoitepohjaisten käyttötapausten etenemisestä kuvasarjat.

Menetelmä vaatii selvästi enemmän työtä opiskelijaprojekteissa kuin perinteinen vaatimusten kerääminen luonnollisella kielellä kuvatuiksi luetteloiksi. Sen avulla voidaan kuitenkin varmistaa samalla kertaa, että ohjelmiston käyttöliittymä vastaa käyttäjän todellisia työnkulkuja ja että järjestelmän vaatimukset ovat muodossa, jota myös asiakas ja loppukäyttäjät ymmärtävät.

4.5 Käyttöliittymän arviointi ja testaus

GUIDen viimeisessä vaiheessa pyritään käyttöliittymäratkaisun testauksen avulla varmistamaan käyttöliittymän hyödyllisyys ja käytettävyys. Arviointia voidaan verrata vesiputousmallin vaatimusmäärittelyn validointivaiheeseen. Molemmissa pyritään

varmistamaan, että järjestelmään tulee kaikki työtehtävien toteuttamiseen tarvittavat toiminnot ja niiden suorittamiseen tarvittava tietosisältö.

Käyttöliittymien arviointiin on kehitetty monenlaisia menetelmiä. Ne voidaan karkeasti jakaa kahteen kategoriaan sen mukaan, tarvitaanko arviointiin testikäyttäjiä. Käyttäjien mukana olon vaativia menetelmiä ovat mm. käytettävyystestaus (usability testing) [Nie93, luku 6], läpikäyntipalaverit (walkthroughs) [Bia91] ja käyttäjätarkkailut (user observations) [Laa06c]. Arviointi voidaan toteuttaa ilman käyttäjää esimerkiksi käyttäen simulointitestausta (usage simulation) [Laa06c], kognitiivista läpikäyntiä (cognitive walkthrough) [LeW97] tai heuristista arviointia (heuristic evaluation) [NiM90]. Yleisesti voidaan sanoa, että menetelmät, joissa käyttäjä on mukana, antavat luotettavampaa tietoa ainakin opittavuusongelmista ja vaativat arvioinnin tekijältä vähemmän asiantuntemusta. Tehokkuusongelmista suurin osa voidaan havaita luotettavasti myös ilman käyttäjää. Ilman käyttäjää toteutettavilla menetelmillä on mahdollista järjestää arviointi myös keskeneräisellä käyttöliittymällä ja niiden järjestäminen on usein mahdollista lyhyemmässä ajassa kuin käyttäjän mukanaoloa vaativien testien [Laa06c].

GUIDessa arviointi toteutetaan käyttäen läpikäyntipalavereita [Bia91], simulointi-testausta ja käytettävyystestejä. Läpikäyntipalaverissa on tyypillisesti mukana käyttöliittymäasiantuntijoita ja järjestelmän käyttäjiä. Palaverissa yksi käyttöliittymä-asiantuntija ohjaa esitystä ja käyttäjät kommentoivat näkemiään työnkulkuja [Bia91]. Läpikäyntipalaverit paljastavat hyvin puutteita ja väärinkäsityksiä työtehtävien kulkujen ymmärtämisessä [Laa04].

Simulointitestauksessa asiantuntija simuloi tavoitepohjaisten käyttötapausten avulla toteutettua käyttöliittymäversiota. Simulointitestauksen eteneminen on kuvattu käyttöliittymät II-kurssin kalvoissa [Laa06c]. Ensimmäiseksi arvioija selvittää käyttäjän parhaan loppuratkaisun käyttötapaukseen, riippumatta arvioitavasta järjestelmästä. Seuraavaksi arvioija selvittää, mikä on järjestelmän tarjoama paras oikea polku tavoitteen saavuttamiseen. Sen jälkeen arvioija simuloi oikean polun useita kertoja. Samalla hän pyrkii selvittämään, mitä toimintoja käyttäjän pitää missäkin vaiheessa tehdä ja mitä mentaalisia toimenpiteitä (kognitiivistä prosessointia) eri vaiheet vaativat. Käyttötapauksen läpikäynnistä muodostetaan kuvasarja. Kuvasarjoista arvioijan on mahdollista havaita mahdolliset ongelmat käyttötapauksen suorituksessa. Havaitut käyttöliittymäongelmat kirjataan erilliseen dokumenttiin. Menetelmä vaatii arvioijalta asiantuntemusta ja kokemusta, mutta paljastaa nopeasti käyttöliittymän suurimmat tehokkuusongelmat [Laa04].

Käytettävyystesti on menetelmä, jolla voidaan tehokkuusongelmien lisäksi havaita myös opittavuuteen liittyviä ongelmia. Menetelmässä käyttäjä suorittaa järjestelmällä hänelle annettuja tehtäviä ja arvioija kirjaa muistiin, miten käyttäjä tehtävät suoritti. Menetelmä on työläs, mutta se tuottaa luotettavaa tietoa käyttöliittymän käytettävyydestä. Käytettävyystesti voidaan tehdä paperiprototyypillä tai valmiilla ohjelmistolla.

Yhteenvetona arviointimenetelmillä voidaan GUIDessa varmistaa ennen suunnittelu- ja toteutusvaihetta, että laaditussa käyttöliittymässä on työnkulkuihin tarvittavat toiminnot ja että niiden suorittaminen käyttöliittymällä on sujuvaa. Näin vähennetään suunnittelu- ja toteutusvaiheessa tulevien muutosten määrää, mikä perinteisessä vaatimus-määrittelyssä on suuri ongelma.

5. Tutkimuskysymykset

Perinteisessä vaatimusmäärittelyssä on heikkoutensa, ja uusissa menetelmissä on piirteitä, joiden avulla myös vesiputousmalliin on mahdollista saada liitettyä järkevä käyttöliittymäsuunnittelu, ja sitä kautta voitaisiin parantaa vaatimusmäärittelyvaihetta. Tässä työssä pyrkimyksenä on selvittää tutkimalla Helsingin yliopiston ohjelmisto-tuotantoprojekteja, kiinnittääkö perinteinen vesiputousmallin vaatimusmäärittely liiaksi myöhemmin tehtäviä käyttöliittymäratkaisuja.

Projektien käyttämässä vesiputousmallissa vaatimuksiin kirjataan tarkkoja kuvauksia siitä, miten ohjelman tulisi toimia. Tällöin saatetaan samalla tulla sitoneeksi käyttöliittymäratkaisuja, koska käyttöliittymä suunnitellaan vaatimusten pohjalta. Ongelmia muodostuu myös silloin, kun lopullisen käyttäjän työnkulkuja ei ole selvitetty, vaan vaatimukset ovat tulleet asiakkaalta. Tällöin käyttöliittymästä voi jäädä pois käyttötilanteiden suorittamisessa tarvittavaa toiminnallisuutta tai tietosisältöä.

Luvuissa 5.1 ja 5.2 käsitellään tämän työn tutkimuskysymykset. Luvussa 5.3 esitellään Helsingin yliopiston ohjelmistotuotantoprojektien rakenne, ja tähän työhön valitut opiskelijaprojektit sekä yleisellä tasolla valittujen opiskelijaprojektien vaatimus-määrittelyvaihe.

5.1 Vaatimusten vaikutukset käyttöliittymäsuunnitteluun

Ensimmäiseksi pyritään selvittämään, onko sellaisia vaatimuksia tai käyttötapauksia, jotka sitovat myöhemmin tehtävän käyttöliittymäsuunnittelun ratkaisuja. Erityisesti tutkitaan, minkälaisia vaatimuksia ne ovat sekä mitkä niistä ovat vahingollisia ja mitkä vaarattomia käyttöliittymälle. Valittujen projektien vaatimusmäärittelyistä kerätään luettelo niistä vaatimuksista, jotka saattavat sitoa käyttöliittymää. Tällaisia ovat mm. ne vaatimukset tai käyttötapaukset, joissa mainitaan jotain käyttöliittymän tekniseen toteutukseen viittaavaa, esimerkiksi ”painaa painiketta”, ”valitsee valikosta”, ”siirtyy sivulle” tai ”selailee listaa”.

Tämän jälkeen käyttöliittymä testataan. Menetelmänä käytetään simulointitestausta, joka paljastaa tehokkuusongelmat sekä mahdollisen puuttuvan tietosisällön ja toiminnallisuuden [Laa04]. Menetelmässä käytetään testitapauksina tavoitepohjaisia käyttötapauksia, jotka pohjautuvat käyttäjän työnkulkuihin. Käyttötapauskuvaukset selvitetään haastattelemalla ohjelmiston todellisia käyttäjiä. Testauksen tuloksena saadaan luettelo käyttöliittymän ongelmakohdista.

Seuraavaksi selvitetään, löytyykö havaittujen käyttöliittymäongelmien ja käyttöliittymää sitovien vaatimusten väliltä yhteys. Tehokkuusongelmaksi voitaisiin esimerkiksi havaita turha navigointi sivulta toiselle ja takaisin. Vaatimusmäärittelydokumentista kerätystä vaatimuslistasta voisi löytyä samaan toiminnallisuuteen liittyvä vaatimus, jossa sanotaan selvästi, että ”siirrytään uudelle sivulle”. Tässä tilanteessa tehokkaampi käyttöliittymäratkaisu voisi olla se, että sivulla olevat tiedot sijoitettaisiin samalle sivulle, jolloin turhalta navigoinnilta vältyttäisiin. Tällöin vaatimus, jossa mainitaan siirtyminen sivulta toiselle, on johtanut kyseiseen huonoon käyttöliittymäratkaisuun. Tämä ei vielä todista sitä, että jokin muu tapa toteuttaa vaatimusmäärittely ei tuottaisi samaa ongelmaa, mutta se osoittaa vesiputousmallissa käytetyn tavan heikkouden. Testauksen jälkeen voidaan myös analysoida, millaiset vaatimukset aiheuttavat riskin huonoihin käyttöliittymäratkaisuihin, ja tarkentaa käsitystä nykyisen vaatimusmäärittelydokumentin ongelmakohdista käyttöliittymäsuunnittelun näkökulmasta.

5.2 Käyttötilanteiden puuttumisen vaikutukset

Toisena tutkimuskysymyksenä selvitetään, aiheuttaako perinteisellä tavalla tehty vaatimusmäärittely käyttöliittymään selviä tehokkuusongelmia ja jääkö käyttöliittymästä puuttumaan käyttäjän käyttötilanteiden kannalta tarpeellista toiminnallisuutta tai tietosisältöä.

Simulointitestaus perustuu käyttäjän todellisiin työtehtäviin. Mikäli sen tuloksena havaitaan käyttöliittymässä edellä kuvattuja ongelmia, voidaan todeta, että opiskelija-projektien tapa toteuttaa vaatimusmäärittely ja sen pohjalta tehtävä käyttöliittymä tuottavat kyseisiä ongelmia. Lisäksi voidaan tällöin suurella todennäköisyydellä epäillä, että projektin alussa toteutettu käyttöliittymäsuunnitelma saattaisi estää kyseiset ongelmat, koska se perustuu samoihin tavoitepohjaisiin käyttötapauksiin, joita simulointitestauksessa käytetään testitapauksina.

Lisäksi työssä selvitään, aiheuttaako nykyinen vaatimusmäärittely käyttöliittymään turhaa toiminnallisuutta tai tietosisältöä. Tämä ei kuitenkaan liene yliopiston opiskelijoiden ohjelmistoprojekteissa kovinkaan yleistä johtuen ohjelmistojen pienestä koosta. Mikäli kuitenkin havaitaan ryhmän suunnittelemassa käyttöliittymässä sellaista toiminnallisuutta, jolle ei löydetä mitään käyttötilannetta, tullaan myös tähän kysymykseen ottamaan kantaa.

Tutkimuskysymyksen tarkoituksena on osoittaa, kuinka oleellista on saada todelliset käyttötilanteet selville vaatimusmäärittelyn kartutusvaiheessa, jotta vaatimukset vastaisivat mahdollisimman hyvin käyttäjän todellisia käyttötilanteita.

5.3 Arvioitavat opiskelijaprojektit

Helsingin yliopiston tietojenkäsittelytieteen laitoksella järjestetään lukukausittain 5-10 ohjelmistotuotantoprojektia. Projekti on perusopintojen pakollinen kurssi. Projektin laajuus on 6 opintoviikkoa ja se kestää n. 14 viikkoa. Kurssilla opiskelijat toteuttavat ohjelmistoprojektin asiakkaalle. Projektien aiheet ovat ennalta määrätty, ja jokainen ilmoittautunut opiskelija voi esittää toiveen, mihin projektiin hän haluaa osallistua. Lopullisen ryhmäjaon tekevät kurssin vastuuhenkilöt.

Kurssiin osallistuvilla opiskelijoilla on oltava suoritettuna pakolliset perusopinnot. Täten kurssin osallistujilla on riittävä taitotaso toteuttaa projekti. Osalle projekti voi olla ensimmäinen suurempi projekti, kun taas työelämässä jo oleville opiskelijoille projekti voi hyvinkin tuntua suppealta. Oppilaiden taitotaso vaihtelee projektiryhmästä toiseen.

Toteutettavat ohjelmistot ovat tyypillisesti hyvin määriteltyjä ohjelmiston osia, joita jokin laitoksen tutkimusryhmä tarvitsee, tai pieniä ohjelmistoja, joita kehitetään esimerkiksi jonkin laitoksen kurssin opetuksen tueksi. Useissa tapauksissa ohjelmistot koostuvat tietokannasta ja käyttöliittymästä, jolla tietokannassa olevaa tietoa voidaan syöttää ja tarkkailla [Tai06].

Projekteja ei vaadita toteutettavaksi millään tietyllä prosessimallilla, mutta suositeltavia prosessimalleja ovat vesiputousmalli ja spiraalimalli [Tai06]. Suurin osa projekteista toteuttaa projektin käyttäen vesiputousmallia tai ainakin hyvin lähellä sitä olevaa mallia. Yksi syy tähän voi olla se, että projektiryhmien on toteutettava projektin aikana tietyt dokumentit, joten esimerkiksi ketterien prosessimallien soveltaminen saattaa olla vaikeaa. Vaadittavia dokumentteja ovat projektisuunnitelma, vaatimusmäärittely-dokumentti ja suunnitteludokumentti. Projektiin osallistuvat opiskelijat vastaavat itse projektin etenemisestä ja valitun prosessimallin noudattamisesta. Jokaisella projektilla on ohjaaja, joka toimii eräänlaisena laadunvalvoja. Ohjaaja voi tarvittaessa suositella tiettyä lähestymistapaa projektin aikana ilmenneisiin ongelmiin, mutta hänen pääasiallinen tehtävänsä on vain seurata, miten projekti etenee ja tarkistaa projektien tuotokset projektin päätyttyä.

5.3.1 Esimerkkiprojektien valintakriteerit

Tämän työn opiskelijaprojektien valinnassa on käytetty seuraavia kriteereitä: Projektin tulee olla päättynyt ja lopputuloksen pitää olla toimiva ohjelmisto, jotta sen tutkiminen on mahdollista. Lisäksi toteutuneella ohjelmistolla tulee olla graafinen käyttöliittymä. Projektien asiakkaan ja tulevien käyttäjien tulisi olla tavoitettavissa, jotta todellisten työnkulkujen selvittäminen olisi mahdollista.

Nämä kriteerit täyttäviä projekteja löytyy lukuisia. Tähän työhön valitut projektit täyttävät edelliset vaatimukset ja ovat projektien osallistujien mukaan onnistuneita projekteja. Tärkein valintaan vaikuttava tekijä oli se, että projektien ohjelmistojen loppukäyttäjät ja asiakas ovat Helsingin yliopiston tietojenkäsittelytieteen laitoksen henkilökuntaa, jolloin myös ohjelmiston käyttäjät, asiakas ja käyttöympäristö ovat tavoitettavissa.

5.3.2 Tutkittavat ohjelmistotuotantoprojektit

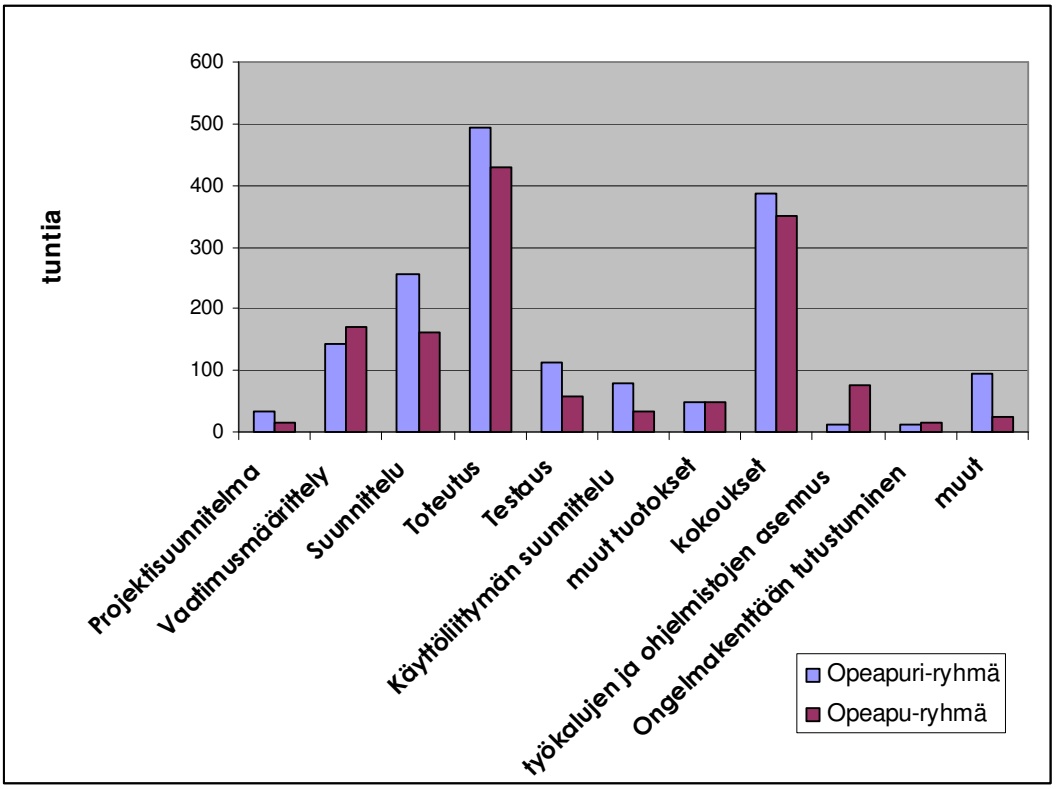
Työssä tutkittavat opiskelijaprojektit ovat Opeapuri ja Opeapu. Molemmat toteutettiin keväällä 2007 Helsingin yliopiston tietojenkäsittelytieteen laitoksella. Molemmissa ryhmissä oli 6 opiskelijaa. Ryhmien asiakkaana toimi yliopistonlehtori Juha Taina. Projektien kesto oli 14 viikkoa.

Projekteilla oli sama aihekuvaus (liite 1). Aiheena oli toteuttaa laitoksen opettaja-tuutorointi-kurssille tapaamisten hallintajärjestelmä. Opetuutoroinnissa uusille oppilaille kootaan ryhmät, joissa seurataan opintojen etenemistä. Kurssiin kuuluu lukukausittain pidettäviä henkilökohtaisia tapaamisia ja ryhmätapaamisia. Työn kuvauksessa vaaditaan ohjelmistolta ainakin seuraavia ominaisuuksia:

- *Ohjelmistolla tulee voida hallita kurssin aikana sovittavia tapaamisia, ryhmätapaamisia ja siinä tulee olla mahdollisuus kirjata tapaamisissa käydyt asiat muistiin.*
- *Opettajatuutori tarvitsee myös nimilistan ja tavan kirjata läsnäolot ylös.*
- *Ohjelmisto tulee toteuttaa www-selaimelle.*

Ryhmät käyttivät prosessimallinaan vesiputousmallia. Opiskelijat pitivät koko projektin ajan tuntikirjanpitoa. Tunneista kuului kirjata määrä ja projektin vaihe. Erillisiksi vaiheiksi on tuntikirjanpidossa eroteltu mm. käyttöliittymän tekeminen. Kuvassa 5 on ryhmien tuntikirjanpidot. Tuntikirjanpidoista käy ilmi, että Opeapuri-ryhmä käytti kokonaisuudessaan 1675 työtuntia, kun taas Opeapu-ryhmä käytti 1384 tuntia. Vaatimusmäärittelyyn Opeapuri-ryhmä käytti n. 8,5 % (142,5 h) kokonaistuntimäärästä, kun taas Opeapu-ryhmä käytti n. 12,4 % (172 h). Tuntikirjanpitoa voidaan pitää suuntaa-antavana, kun verrataan projektien vaiheiden kokoa. Tämän pohjalta voidaan sanoa,

että Opeapu-ryhmä on käyttänyt sekä suhteellisesti että määrällisesti enemmän aikaa vaatimusmäärittelyvaiheeseen. Käyttöliittymäsuunnitteluun Opeapuri-ryhmä on käyttänyt 4,8 % (80 h) kokonaistuntimäärästä, kun Opeapu-ryhmä käytti vain 2,4 % (33,5 h). Tuntikirjanpidon mukaan Opeapuri-ryhmä on panostanut enemmän käyttöliittymäsuunnitteluun.



Kuva 5. Tuntikirjanpidot.

5.3.3 Projektien vaatimusmäärittelyvaihe

Projektit toteuttivat vesiputousmallin mukaisen vaatimusmäärittelyvaiheen. Sen tuloksena ryhmät tuottivat vaatimusmäärittelydokumentin [Ekl07, Tuo07]. Molemmat ryhmät kirjasivat vaatimusmäärittelydokumenttiin käyttötapaukset, käyttäjävaatimukset, järjestelmävaatimukset, tietosisältökaavion ja yksinkertaisen käyttöliittymän sivujaon. Lisäksi Opeapuri-ryhmä toteutti jo tässä vaiheessa alustavia käyttöliittymäkuvia, joita ei kuitenkaan liitetty vaatimusmäärittelydokumenttiin. Opeapuri-ryhmä kirjasi lisäksi muistiin toimintoympäristövaatimukset. Opeapu-ryhmä kirjasi nämä vaatimukset ei-toiminnallisiin järjestelmävaatimuksiin.

Käyttötapauksista molemmat ryhmät kirjasivat, kuka tehtävän saa tehdä (käyttäjärooli), lyhyen kuvauksen siitä miten, käyttötapauksen toiminta etenee, sekä mahdolliset poikkeustilanteet. Opeapuri-ryhmä kirjasi lisäksi käyttötapausten prioriteetit, mutta Opeapu-ryhmä kirjasi prioriteetit vain käyttäjä- ja järjestelmävaatimuksiin. Opeapu-ryhmän käyttötapauksissa oli alkutila ja lopputila. Alkutilalla pyritään dokumentissa kuvaamaan tarvetta, joka on johtanut käyttötapaukseen alkamiseen. Lopputila voidaan ajatella käyttötapauksen suorittamisen lopputuloksena. Taulukoissa 1 ja 2 on esitetty molempien ryhmien versio samasta käyttötapauksesta. Opeapu-ryhmä on jaotellut käyttötapaukset käyttäjäroolien mukaan, joten käyttäjärooli ei ole näkyvillä kuvassa, vaan se on aliluvun otsikkona.

Käyttötapaus	**Opiskelijan ryhmän vaihtaminen**
Käyttäjän rooli	Super-user, opetuutori
Käyttäjän toiminta	Avaa ryhmienselaussivun (super-user valitsee ensin opetuutorin). Valitsee siirrettävän opiskelijan listasta sekä ryhmän, johon opiskelija siirretään ja painaa siirtämisnappia.
Järjestelmän toiminta	Vaihtaa tietokannassa opiskelijan ryhmä-kentän tiedon.
Poikkeustilanne	Opiskelijan uusi ryhmä on sama kuin vanha. Tietokantapalvelimeen ei saada yhteyttä. Opiskelijaa ei ole valittuna.
Prioriteetti	3: Pyritään toteuttamaan

Taulukko 1. Opeapuri-projektin versio käyttötapauksesta [Ekl07].

Opiskelijan siirtäminen toiseen tuutoriryhmään

Alkutila	Opettajatuutori haluaa siirtää opiskelijan omasta tuutoriryhmästään johonkin toiseen.
Lopputila	Opiskelija on siirretty tuutoriryhmästä toiseen.
Kuvaus	Ylläpitäjä siirtyy ryhmien hallinnan välilehdelle. Sitten hän valitsee siirrettävän opiskelijan sekä ryhmän, johon opiskelija siirretään ja siirtää opiskelijan.
Poikkeukset ja rajoitukset	-

Taulukko 2. Opeapu-projektin versio käyttötapauksesta [Tuo07].

Molemmat ryhmät kirjasivat käyttäjävaatimukset luonnollisella kielellä kuvattuina luetteloina. Vaatimukset on dokumenteissa jaettu toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. Lisäksi molemmat käyttävät neljän tason priorisointia. Tällä pyritään varmistamaan, että ainakin tärkeimmät toiminnot tulevat toteutetuiksi.

Järjestelmävaatimusten kirjaamisessa on dokumenteissa selviä eroja. Opeapu-ryhmä pyrkii järjestelmävaatimuksissa kuvaamaan käyttäjävaatimukset astetta tarkemmalla tasolla kuin Opeapuri-ryhmä. Esimerkiksi käyttäjävaatimusta *”ylläpitäjä voi lisätä uusia käyttäjiä järjestelmään”* vastaava järjestelmävaatimus on ”*Uusia käyttäjiä rekiste-*

röitäessä on annettava käyttäjän nimi, käyttäjätunnus ja sähköpostiosoite”. Jaottelulla on ilmeisesti pyritty pitämään käyttäjävaatimukset sellaisessa muodossa, että myös asiakkaan olisi mahdollista ymmärtää ne, ja sitten järjestelmävaatimuksissa on pyritty kertomaan tarkemmin järjestelmän kannalta tarpeellisia vaatimuksia. Opeapuri-ryhmän vaatimusmäärittelydokumentissa on järjestelmävaatimuksia vain viisi kappaletta. Ryhmä on ilmeisesti katsonut käyttäjävaatimusten riittävän ja kirjannut järjestelmävaatimuksiin vain yleisellä tasolla tapoja, miten järjestelmän tulisi käyttäytyä. Järjestelmä-vaatimuksissa mainitaan esim. automaattinen uloskirjautuminen 2 tunnin kuluttua sivun latauksesta.

Dokumentteihin kirjatut käyttöliittymän sivukartat olivat molemmissa hyvin samantyylisiä. Molemmissa kuvataan sivujako ja mahdolliset siirtymät sivulta toiselle. Tarkempaa käyttöliittymäkuvausta ei kummassakaan dokumentissa ollut.

6. Tutkimuksen toteutus ja tulokset

Tämän työn ensimmäisessä vaiheessa arvioitiin, kuinka paljon projektien vaatimusmäärittelydokumentissa kirjatut eritasoiset vaatimukset ja käyttötapaukset sitovat käyttöliittymäratkaisuja (luku 6.1). Toisessa vaiheessa selvitettiin opettajatuutorien nykyisiä työnkulkuja ja niiden takana olevia tavoitteita. Menetelmänä käytettiin luvussa 4.3.2 esiteltyä kontekstuaalista käyttäjähaastattelua. Selville saatujen käyttäjän tavoitteiden ja työnkulkujen pohjalta luotiin kolme keskeisintä tavoitepohjaista käyttötapausta (luku 6.2). Seuraavassa vaiheessa opiskelijaryhmien toteuttamille käyttöliittymille tehtiin simulointitestaus käyttäen sen syötteenä luotua tavoitepohjaista käyttötapausta (luku 6.3 ja 6.4). Luvuissa käsiteltävät dokumentit ovat Opeapuri-ryhmän vaatimusmäärittelydokumentti [Ekl07] ja Opeapu-ryhmän vaatimusmäärittelydokumentti [Tuo07].

6.1 Vaatimusten analysointi

Analyysivaiheessa tärkein tavoite oli löytää ne vaatimukset ja käyttötapaukset, jotka vaikuttavat tulevaan käyttöliittymäsuunnitelmaan. Vaiheessa pyrittiin löytämään vaatimuksia, jotka sitovat toimintoja ja tietosisältöjä, sekä vaatimuksia, jotka sitovat toimintojen interaktiotapoja käyttöliittymässä. Analyysin aikana kysymykset, kuten sitooko vaatimus tietosisältöä, toimintoja tai käyttöliittymän interaktiotapoja, olivat keskeisimpiä. Tässä vaiheessa ei vielä mietitty, aiheuttavatko havainnot käyttöliittymään käytettävyysongelmia.

6.1.1 Käyttötapaukset

Käyttötapauksia on Opeapuri-ryhmän vaatimusmäärittelydokumentissa 31, ja Opeapu-ryhmän dokumentissa niitä on 42. Tärkein havainto dokumenteista oli, että lähes jokainen käyttötapaus sitoo toiminnallisuutta, tietosisältöä ja käyttöliittymän rakennetta ja interaktiotapoja. Käyttöliittymäratkaisuja kiinnittäviä ratkaisuja on kahden tasoisia. Toiset kiinnittävät toiminnon, mutta eivät toiminnon suorittamisen interaktiotapoja. Toiset taas kiinnittävät toiminnon lisäksi myös käyttöliittymäratkaisun interaktiotavankin yksityiskohtaisesti [Ket07]. On tietenkin ymmärrettävää, että käyttötapaus sitoo toiminnallisuutta ja tietosisältöä, koska sen luonne on kuvata tietyn toimintoketjun eteneminen. Samasta syystä usein myös käyttöliittymäkomponentit ja niiden toimintalogiikka saattaa tulla kuvattua hyvinkin tarkasti.

Esimerkiksi taulukossa 3 kuvatussa käyttötapauksessa määritellään, että käyttöliittymässä on oltava erillinen välilehti, josta käyttäjä näkee ryhmän jäsenet. Näin sidotaan käyttöliittymän rakennetta. Parempaa kuvauskieltä on kohdassa, jossa mainitaan ”josta pääsee tilaan, jossa opiskelijan henkilötietojen muokkaaminen on mahdollista”. Tässä kohdassa ei sidota sivua, jolla muokkaus tapahtuu, vaan käyttöliittymäsuunnittelija voi itse päättää, sijoittaako hän muokkauksen samalle vai erilliselle sivulle. Tämäkin päätelmä riippuu toisaalta siitä, miten suunnittelija ymmärtää tilan käsitteen. Jos tilana voidaan käsittää nykyinen sivu, ilmaisu on parempi.

Opiskelijan tietojen muuttaminen

Alkutila	Opettajatuutori haluaa muuttaa oman tuutoriryhmänsä opiskelijoiden henkilötietoja.
Lopputila	Opettajatuutori on saanut tehtyä haluamansa muutokset omaan tuutoriryhmäänsä ja on ”opiskelijan tiedot” -näkymässä.
Kuvaus	Opettajatuutori siirtyy siihen järjestelmän välilehteen, josta hän näkee omaan tuutoriryhmäänsä kuuluvien opiskelijoiden nimet. Opettajatuutori valitsee haluamansa opiskelijan tuutoriryhmästään painamalla opiskelijan vieressä olevaa valintanappia. Tämän jälkeen opettajatuutori painaa ”opiskelijan tiedot” -nappia, josta hän pääsee tilaan jossa opiskelijan henkilötietojen muokkaaminen on mahdollista. Opettajatuutori tekee tarvittavat muutokset opiskelijan tietoihin sekä kuittaa muutokset painamalla ”talleta muutokset” -nappia.
Poikkeukset ja rajoitukset	Opiskelijan käyttäjätunnusta ei voida muuttaa sellaiseksi joka on jo jollain järjestelmässä olevalla käyttäjällä.

Taulukko 3. Opeapu-ryhmän käyttötapaus, joka kiinnittää käyttöliittymäratkaisun interaktiotapoja [Tuo07].

Taulukon 4 käyttötapauksessa on taas kuvattu koko toimintoketju sivun avaamisesta siihen, että käyttäjä painaa Tallenna-painiketta. Käytännössä siinä on suunniteltu koko käyttötapauksen kulku käyttöliittymässä kohta kohdalta. Käyttöliittymäsuunnitteluvaiheessa ei siis voida enää muuta kuin sijoitella komponentit paikoilleen. Käyttötapaus kiinnittää toimintojen lisäksi myös käyttöliittymäratkaisun.

Käyttötapaus	**Opiskelijoiden lisääminen**
Käyttäjän rooli	Opetuutori
Käyttäjän toiminta	Avaa opiskelijanlisäyssivun. Syöttää tyhjiin kenttiin opiskelijan nimen (pakollinen tieto), käyttäjätunnuksen (pakollinen tieto, ohjelma voi myös generoida tunnuksen annetusta nimestä käyttäjän painaessa generointinappia), sähköpostiosoitteen, salasanan (voi jättää tyhjäksi) ja tarvittaessa vapaamuotoista tekstitietoa opiskelijasta, ja valitsee opiskelijalle ryhmän listasta (yhden omista ryhmistään.), ja painaa lisäysnappia.
Järjestelmän toiminta	Jos käyttäjä painaa generointinappia, järjestelmä generoi tunnusehdotuksen annetusta oppilaan nimestä. Tallentaa syötetyt tiedot tietokantaan ja luo uuden käyttäjän.
Poikkeustilanne	Pakollinen syötettävä tieto (nimi tai tunnus) puuttuu. Saman tunnuksen

Taulukko 4. Opeapuri-ryhmän sitova käyttötapaus [Ekl07].

Käyttötapaukset vaikuttavat kiinnittävän hyvin paljon tulevan käyttöliittymäsuunnittelijan mahdollisuuksista suunnitella sivuja, joilla toiminnot tapahtuvat. Käyttötapaukset on usein kuvattu niin tarkalla tasolla, että niiden kirjoittaja on jo käytännössä joutunut ainakin mielessään käymään läpi, miten toiminto käyttöliittymässä etenee. Käytännössä käyttötapauksen kirjoittaja on huomaamattaan vaatimustenmäärittelyvaiheessa jo suunnitellut toiminnon kulun myöhemmin suunniteltavassa käyttöliittymässä.

6.1.2 Käyttäjävaatimukset

Toiminnallisia käyttäjävaatimuksia on Opeapuri-ryhmän dokumentissa 28 [Ekl07] ja Opeapu-ryhmällä niitä on 31 [Tuo07]. Toiminnallisissa vaatimuksissa on selvästi vähemmän suoria viittauksia käyttöliittymäkomponentteihin ja niiden toimintalogiikkaan kuin käyttötapauksissa. Merkittävin yksittäinen käyttäjävaatimuksissa esiintyvä käyttöliittymäkomponentti on kalenteri. Vaatimus varausten käsittelemisestä kalenterin kautta oli jo aihekuvauksessa. Se lieneekin suurin syy miksi vaatimuksissa oletetaan, että käyttöliittymässä tulee olemaan kalenteri. Opeapu-ryhmä on kirjannut vaatimuksen kalenterista suoraan ei-toiminnallisiin vaatimuksiin. Tässä vaiheessa ei voida kuitenkaan olla varmoja, onko kalenteri paras ratkaisu kaikkiin varaamiseen liittyviin toimintoihin. Tässä on siis tehty käyttöliittymän rakennetta koskeva päätös.

Vaatimuslistoja tutkiessa herää monessa kohdassa kysymys miksi. Esimerkiksi Opeapu-ryhmän vaatimuksessa 22 mainitaan että: ”Opettajatuutori näkee järjestelmästä muiden opettajatuutoreiden nimet”. Lukija ymmärtää kyllä vaatimuksen, mutta vaatimus ei itsessään kerro tilannetta, jossa sitä on ajateltu tarvittavan. Vaatimuslista kuvaa, mitä toimintoja ja osaksi myös mitä tietosisältöä toiminnon toteuttamiseen vaaditaan, mutta jättää vastaamatta, missä tilanteissa kyseisiä toimintoja tarvitaan.

Käyttäjävaatimuslistan jokainen vaatimus vastaa käytännössä yhtä toimintoa. Se ei kuitenkaan sido sitä, miten toiminto tulisi järjestelmän käyttöliittymässä toteuttaa, vaan vain ilmoittaa, että tällainen toiminto on siellä oltava. Toisaalta se ei myöskään ota kantaa siihen, liittyvätkö toiminnot toisiinsa tai onko tarpeen käyttöliittymäsuunnittelussa huomioida tiettyjen toimenpiteiden toteutusjärjestystä.

Ryhmien kirjaamistyylissä on selvä ero. Opeapuri-ryhmä on pyrkinyt kirjaamaan käyttäjävaatimukset selvästi korkeammalla tasolla kuin Opeapu-ryhmä. Taulukoissa 5 ja 6 on ryhmien versiot vaatimuksesta ”mahdollistaa muistiinpanojen kirjaus”. Opeapuri-ryhmä kuvaa yhdessä vaatimuksessa saman, minkä Opeapu-ryhmä kuvaa vaatimuksilla

11,13,14 ja 15. Korkeammalla tasolla esitetty vaatimus antaa paremman kuvan toiminto-kokonaisuudesta verrattuna pilkottuun versioon samasta asiasta.

Tapaamisten muistiinpanojen kirjaaminen
Opetuutori voi kirjoittaa vapaamuotoisia muistiinpanoja tapaamisista, ryhmätapaamisesta ryhmäkohtaisesti sekä opiskelijakohtaisesti ja henkilökohtaisesta tapaamisesta opiskelijakohtaisesti. Opiskelijakohtaiseen kenttään voi esimerkiksi kirjoittaa poissaoloon liittyviä lisätietoja.

Taulukko 5 Opeapuri-ryhmän toiminnallinen käyttäjävaatimus [Ekl07].

11. Opettajatuutori voi lisätä lisätietoja jokaiseen yksilötapaamiseen. (1)
12. Opettajatuutori voi muokata ja poistaa yksilötapaamisiin liittyviä lisätietoja. (2)
13. Opettajatuutori voi lisätä kalenteriin ryhmätapaamisaikoja, sekä poistaa niitä sieltä. (1)
14. Opettajatuutori voi lisätä lisätietoja jokaiseen ryhmätapaamiseen. (1)
15. Opettajatuutori voi muokata ja poistaa jokaiseen ryhmätapaamiseen liittyviä lisätietoja. (2)

Taulukko 6. Opeapu-ryhmän toiminnallinen käyttäjävaatimus[Tuo07].

Ei-toiminnallisistakin käyttäjävaatimuksista yllättävän moni sisältää vaatimuksia, jotka kiinnittävät käyttöliittymäratkaisuja. Esimerkiksi Opeapuri-ryhmän taulukossa 7 kuvattu vaatimus, jossa kerrotaan, miten käyttäjän virheitä pyritään estämään. Vaatimuksessa tehdään koko käyttöliittymään vaikuttava päätös piilottaa painikkeet, joita ei sillä hetkellä ole luvallista painaa. Toisaalta ei-toiminnallisiin vaatimuksiin on kirjattu paljon suorituskykyyn ja toimintaympäristöön liittyviä vaatimuksia, jotka eivät kiinnitä käyttöliittymäratkaisua. Opeapu-ryhmän dokumentissa ei ollut erikseen toiminto-ympäristövaatimuksia. Nämä vaatimukset eivät sitoneet käyttöliittymäratkaisuja.

Käyttäjän virhetoimintojen ehkäiseminen
Virhetoimintoja ehkäistään käyttöliittymällä, siten että hävitetään näkyvistä painikkeita silloin, kun niiden painamisesta aiheutuisi virhe.

Taulukko 7. Opeapuri-ryhmän ei-toiminnallinen käyttäjävaatimus.

6.1.3 Järjestelmävaatimukset

Järjestelmävaatimusten määritelmän mukaan niillä pyritään kuvaamaan käyttäjä-vaatimukset sillä tarkkuudella, että järjestelmän toteutus niiden pohjalta on mahdollista

[Som01, s. 109]. Kummankaan projektin järjestelmävaatimuksia ei ole kirjoitettu näin, vaan ne, ovat ennemminkin yksittäisten tarkennusten listoja. Opeapu-ryhmä kirjasi 28 toiminnallista ja 22 ei-toiminnallista järjestelmävaatimusta. Opeapuri-ryhmän dokumentissa oli vain kaksi toiminnallista ja kolme ei-toiminnallista järjestelmävaatimusta. Opeapuri-ryhmän dokumentissa oli ei-toiminnallisista järjestelmävaatimuksista eroteltu kolme ympäristövaatimusta omaksi osaksi. Lukumäärien suuri ero johtuu kirjaustavan eroista. Opeapuri-ryhmä kirjasi vain yleisellä tasolla muutamia koko järjestelmää koskevia vaatimuksia, kun taas Opeapu-ryhmä kirjasi vaatimukset huomattavasti yksityiskohtaisemmin.

Opeapu-ryhmän järjestelmävaatimuksista sekä toiminnalliset että ei-toiminnalliset ovat luonteeltaan reunaehtoja ja rajoituksia. Taulukossa 8 on muutamia esimerkkejä tämäntyyppisistä vaatimuksista. Järjestelmävaatimukset eivät sisällä varsinaisesti uutta toiminallisuutta, vaan ne ennemminkin tarkentavat toimintoihin tarvittavaa tietosisältöä. Järjestelmävaatimuksista ei ole myöskään viitettä käyttäjävaatimuksiin. Näin tarkentavan tiedon ja käyttäjävaatimuksen yhteyden hahmottaminen on dokumentista vaikeaa.

37. Uusia käyttäjiä rekisteröitäessä on annettava käyttäjän nimi, käyttäjätunnus ja sähköpostiosoite. (1)
38. Ylläpitäjä voi lisätä opiskelijan mihin tahansa tuutoriryhmään. (1)
39. Ylläpitäjä voi siirtää opiskelijoita tuutoriryhmästä toiseen. (1)
40. Ylläpitäjä voi asettaa käyttäjän salasanan oletusarvoonsa, joka on aina aluksi sama kuin käyttäjän käyttäjätunnus. (1)

Taulukko 8. Opeapu-ryhmän järjestelmävaatimuksia [Tuo07].

Järjestelmävaatimuksissa ei kiinnitetä käyttöliittymän rakenteeseen liittyviä ratkaisuja, mutta yksittäisten sivujen sisältöön vaatimuksissa on hyvinkin tarkkoja kuvauksia. Taulukossa 9 on muutamia esimerkkejä. Osa on hyvinkin tarkkoja määritelmiä siitä, miten asiat tulee esittää, kuten vaatimukset kalenterin värityksestä ja tuntijaosta.

42. Opettajatuutori saa järjestelmästä tuutoriryhmäänsä liittyvän tulosteen, josta selviävät
    - opiskelijoiden läsnä- ja poissaolot tapaamisissa, (1)
    - opiskelijoiden nimi, käyttäjätunnus ja sähköpostiosoite (2)
    - sekä opiskelijat, jotka eivät ole varanneet aikaa yksilötapaamiseen. (3)
71. Kalenterinäkymässä on eroteltu väreillä varatut ja varaamattomat yksilötapaamisajat.
78. Kalenterimerkinnät ovat puolen tunnin tarkkuudella ja varattavat ajat alkavat joko tasatunnein tai puolelta.

Taulukko 9. Opeapu-ryhmän järjestelmävaatimuksia, joka sitovat käyttöliittymäratkaisuja [Tuo07].

Opeapuri-ryhmän järjestelmävaatimuksissa kiinnitettiin hyvin vahvasti käyttöliittymäratkaisuja. Taulukossa 10 on kaksi hyvin vahvasti mutta hieman eritavoilla käyttöliittymäratkaisuja kiinnittävää esimerkkiä. Ensimmäinen kiinnittää koko käyttöliittymää koskevan tavan antaa käyttäjälle palautetta. Toinen taas kiinnittää hyvin tarkasti, miltä käyttöliittymään tulevan kalenterin pitäisi näyttää.

Virheilmoitukset ja käyttäjälle annettava palaute

Käyttäjän käyttäessä järjestelmää tietokantaa sekä käyttöliittymänäkymää päivitetään käyttäjän toimintojen mukaisesti kolmen minuutin kuluessa. Mikäli tietokannan päivitys ei onnistu tässä ajassa, järjestelmä antaa virhettä vastaavan virheilmoituksen. Mikäli toiminto onnistuu, järjestelmä ilmoittaa siitä käyttäjälle päivittämällä käyttöliittymänäyttöä uutta tilannetta vastaavaksi, tai erillisellä onnistumisilmoituksella, mikäli tietokannan päivitys ei aiheuta muutoksia käyttöliittymänäkymään.

Kalenteri

Kalenteri sisältää ajat maanantaista perjantaihin klo 08-20. Päivät on jaettu tasaisesti 30 minuutin pituisiin osiin.

Taulukko 10. Opeapuri-ryhmän käyttöliittymäratkaisuja kiinnittäviä järjestelmävaatimuksia [Ekl07].

6.2 Testitapausten luonti

Toisessa vaiheessa tutkimusta luotiin simulointitestauksen syötteeksi tarvittavat tavoitepohjaiset käyttötapaukset. Selvittely aloitettiin haastattelemalla järjestelmän tulevaa todellista käyttäjää (luku 6.2.1). Haastattelun tulosten pohjalta luotiin sitten järjestelmän keskeisimmät tavoitepohjaiset käyttötapaukset (luku 6.2.2), joiden pohjalta seuraavassa vaiheessa simulointitestaus voitiin toteuttaa.

6.2.1 Kontekstuaalinen käyttäjähaastattelu

Käyttöliittymän simulointitestauksessa tarvittavat käyttötilanteet syntyivät kontekstuaalisen käyttäjähaastattelun pohjalta. Haastateltavana oli laitoksen opettaja-tuutoroinnista vastaava henkilö, joka toimii itse myös opettajatuutorina. Haastattelu kesti tunnin. Sen aikana käsiteltiin vastuuhenkilön työnkulkuja ja opettajatuutorin työnkulkuja. Huomiota kiinnitettiin erityisesti opettajatutorin työskentelytapoihin. Haastattelun aikana kirjattiin lyhyesti paperille keskeisimpiä havaintoja, ja haastattelun jälkeen kaikki keskeisimmät havainnot kirjattiin tarkemmin tekstitiedostoon.

Haastattelussa käytiin yksityiskohtaisesti läpi, miten opettajatuutorin työnkulut menevät. Erityisesti keskityttiin henkilökohtaisten tapaamisten sopimiseen, koska se on kuitenkin tässä työssä tutkittavien projektien keskeisin toiminnallisuus. Haastattelun aikana käytiin läpi, miten opettajatuutori oli edellisen opiskelijaryhmänsä kanssa toiminut. Esimerkkitilanteena käsiteltiin ryhmän ensimmäisen lukukauden aikana tapahtuvat tilanteet, joista keskeisimpänä oli se, miten ensimmäiset henkilökohtaiset tapaamiset sovitaan. Muita esiin tulleita tilanteita olivat ryhmätapaamiset ja niitä ennen, niiden aikana ja niiden jälkeen tehtävät toimenpiteet. Suurin osa tapahtumista käsiteltiin niin, että opettajatuutori kertoi, miten hän missäkin tilanteessa nykyään toimii ja mitä ohjelmistoja hän käyttää apuna. Haastattelu eteni siten, että haastateltava kertoi hyvin yleisellä tasolla, miten hän missäkin tilanteessa toimi. Haastattelija joutui usein tarkentavien kysymysten avulla kaivelemaan konkreettisia esimerkkejä tilanteista, mikä osoittautui hyvin haastavaksi. Tämä johtui siitä, että on hyvin vaikea keksiä järkeviä esimerkkejä kesken intensiivisen haastattelun. Näin osa käyttötilanteista tuli käsiteltyä vain pintapuolisesti ilman konkreettisia esimerkkejä. Siitä huolimatta haastattelusta sai selkeän kuvan opettajatuutorin työnkuluista.

Haastattelussa selvisi, että kurssi on uudistunut syksyllä 2007 ja sen sisältöä ja toimintatapoja on jonkin verran muutettu sitten viime kevään 2007, jolloin tässä työssä tutkitut opiskelijaprojektit toteutettiin. Opettajatuutoroinnissa ennen pakollisina pidetyt, joka lukukausi järjestettävät ryhmätapaamiset ovat muuttuneet vapaaehtoisiksi. Tilalle on tullut suorituspassi, johon opiskelijan tulee kerätä joka lukukausi vähintään seitsemän merkintää. Merkintöjä saa mm. kaikista opiskeluun liittyvistä tapahtumista, kuten esittelytilaisuuksista, vapaaehtoisista ryhmätapaamisista ja opettajatuutoroinnin henkilökohtaisista tapaamisista. Opettajatuutorilla on siis nykyään vapaammat kädet järjestää ryhmälleen haluamiaan tapahtumia.

Toinen keskeinen havainto oli, että kurssilla on jo useita ohjelmistoja apuvälineenä, keskeisimpinä Kurki 2.0 ja Moodle. Kurki on opetushenkilökunnalle suunniteltu läsnäolojen ja laskuharjoitusryhmien hallinnointijärjestelmä, joka saa ilmoittautuneiden opiskelijoiden tiedot Ilmo-ilmoittautumisjärjestelmästä. Kurki on ohjelmisto, jolla hallinnoidaan myös opettajatuutorointiin osallistuvien opiskelijoiden läsnäoloja suorituspassin lisäksi. Moodle-ohjelmisto on oppimisalusta, jolle on mahdollista rakentaa verkkokursseja tai lähiopetuksen tueksi tarkoitettuja verkko-osioita. Opettaja-tuutoroinnissa Moodleen on luotu informaatiosivu, jonka avulla hoidetaan keskitetysti tiedottaminen, ja sieltä löytyy myös suuri joukko ohjeita kurssin suorittamiseen.

Haastattelussa selvisi, että yhdessä opettajatuutori-ryhmässä on noin 15 opiskelijaa. Tapaamisia on yksi joka lukukausi. Opettajatuutori sopii tapaamiset siten, että hän lähettää oppilaille sähköpostin, jossa hän tarjoaa muutamia vaihtoehtoisia päiviä, jolloin hän voisi tapaamiset pitää. Tämän jälkeen opiskelijat lähettävät ehdotuksensa ajankohdista, jotka sopisivat heille. Mikäli mikään aika ei käynyt, opiskelijalle oli mahdollista sopia tapaaminen myös muille kuin ehdotetuille päiville. Opettajatuutori vahvisti tapaamiset lähettämällä vahvistussähköpostin, jossa hän ilmoitti ehdotetun ajankohdan sopivan. Opettajatuutori kirjasi sovitut ajat omaan kalenteriinsa. Tapaamisen aikana opettajatuutori käy opiskelijan kanssa läpi opiskelijan ennen tapaamista palauttaman eHopsin, joka sisältää erilaisia opiskeluun liittyviä kysymyksiä. Tapaamisesta ei kirjata mitään muistiinpanoja, vaan ainoastaan läsnäolotiedot merkitään suorituspassiin ja Kurki-järjestelmään. Tapaamisen jälkeen opettajatuutori ilmoittaa esiin tulleista epäkohdista, kuten kurssien aikataulutusongelmista suoraan henkilökohtaisesti niistä vastaaville henkilöille, mikäli sellaisia on tapaamisen aikana esille tullut.

Haastattelussa vaikeinta oli ehdottomasti saada haastateltava näyttämään esimerkkien kautta, miten työ etenee, koska haastattelun aikana on vaikea keksiä kysymyksiä, joiden avulla haastattelu etenisi konkreettisten tilanteiden esittämiseen. Vaikka haastattelussa ei joka tilanteessa päästy konkreettiselle tasolle, se antoi kuitenkin todella paljon ymmärtämystä opettajatuutorin työtavoista ja välineistä, joita hänellä jo on käytössä, ja selkeän kuvan opettajatuutorin nykyisistä työskentelytavoista. Keskeisin havainto oli, että opettajatuutorilla on jo välineet, joilla läsnäoloja ja kurssiin liittyvää materiaalia hallitaan, joten näiden toteuttaminen uudestaan työssä tutkittavissa projekteissa vaikuttaa tarpeettomalta. Lisäksi haastattelu antoi hyvin vastauksia siihen, miten henkilökohtaiset tapaamiset tällä hetkellä toteutetaan.

6.2.2 Tavoitepohjaisten käyttötapausten luonti

Kontekstuaalisen haastattelun jälkeen luotiin tavoitepohjaiset käyttötapaukset pohjautuen haastattelussa selvinneisiin työnkulkuihin. Haastattelussa ilmeni, että opettajatuutorilla on jo useita järjestelmiä, joiden avulla hallitaan mm. läsnäoloja. Haastattelun jälkeen tehtiin päätös, että tässä työssä tullaan keskittymään erityisesti henkilökohtaisten tapaamisten sopimiseen liittyviin työnkulkuihin, koska tapaamisten sopiminen on testattavien ohjelmistojen keskeisin toiminnallisuus. Työhön käytettävän ajan rajallisuudesta johtuen luotiin lopulliseen muotoonsa ainoastaan opettajatuutorin keskeisimmät tavoitepohjaiset käyttötapaukset, jotka on kuvattu kuvissa 6, 7 ja 8. Niissä opettajatuutorin tavoite on saada sovittua henkilökohtaiset tapaamiset ryhmään kuuluvien opiskelijoiden kanssa. Tästä eteenpäin puhuttaessa käyttötapauksesta tarkoitetaan tavoitepohjaista käyttötapausta.

Ensimmäinen käyttötapaus on erikoistapaus, koska sen sisällä on kaksi erillistä käyttötapausta, jotka ovat käyttötapaukset kaksi ja kolme. Tämä erikoisuus johtuu siitä, että ensimmäinen käyttötapaus on korkean tason kuvaus koko käyttötilanteesta, jossa toimijoina ovat opettaja ja opiskelijat. Käyttötapaukset kaksi ja kolme aktivoituvat sen päätöksentekokohdissa. Käyttötapaus kaksi aktivoituu päätöksentekokohdassa, jossa ryhmätapaamisen jälkeen opettaja on tehnyt päätöksen, miten hän sopii loput tapaamiset. Kolmas käyttötapaus aktivoituu päätöksentekokohdassa, jossa opettajan on varmistuttava, ovatko opiskelijat varanneet henkilökohtaisen tapaamisen. Käyttötapauksen luonnissa suurin haaste oli muotoilla käyttäjän tavoite ja ongelma sellaiseen muotoon, että se vastaisi mahdollisimman hyvin käyttäjän todellista tavoitetta.

KT 1. Henkilökohtaisten tapaamisten sopiminen

Opettaja Janin tavoite ja ongelma:

Jani on opettajatuutorina laitoksen opetuutorointi-kurssilla. Vastuuhenkilö on lähettänyt Janille sähköpostin, josssa kerrotaan hänen ryhmänsä ja ensimmäisen ryhmätapaamisen ajankohta ja luokka. Kurssilla järjestetään vuosittain henkilökohtaiset tapaamiset, joissa käydään läpi eHopsiin kirjattuja asioita. Maanantaina 15.10.2007 on eHopsin viimeinen palautuspäivä. Jani tietää, että tämän päivän jälkeen hän voi sopia tapaamiset. Jani pitää yleensä tapaamiset muutamana päivänä kootusti, näin hän saa tapaamiset sujuvimmin pidettyä, eivätkä ne häiritse muuta työtä. Ryhmällä on perjantaina 12.10.2007 ensimmäinen ryhmätapaaminen. Jani tietää, mitä nopeammin hän tapaamiset sopii, sen nopeammin hän saa ne pidettyä. Jani ei kuitenkaan tiedä, mikä olisi helpoin tapa sopia tapaamiset. Jani ei myöskään tiedä, mitkä ajat oppilaille sopivat.

Opiskelijan tavoite ja ongelma

Petteri on aloittanut opiskelun tänä syksynä. Petteri on käynyt opettajatuutoroinnin ensimmäisellä kaikille yhteisellä luennolla. Petteri tietää, että kurssi on pakollinen ja, että sen suorittamiseen vaaditaan lukukausittain pidettävä henkilökohtainen tapaaminen. Petteri tietää, että ryhmätapaaminen on 15.10.2007. Hän ei tiedä milloin tai miten henkilökohtaiset tapaamiset pitäisi sopia.

Tilatiedot:

- Nyt on keskiviikko 10.10.2007. Jani on saanut edellisenä päivänä vastuuhenkilöltä listan ryhmänsä oppilaista.
- Jani pitää tapaamiset mielellään 9-16 välillä.
- Janin ryhmään kuuluu 15 oppilasta.
- Janilla seuraavat päivät sellaisia, jolloin hän voisi tapaamiset pitää:
    - to 25.10
    - pe 26.10
    - ma 29.10
    - ti 30.10 vain 10-14
    - to 1.11
    - pe 2.11
- Jani tietää, että opetuutorille on tarjolla ohjelmisto, jolla varaukset voisi tehdä.
- Jani on hoitanut tapaamisten sopimisen aina sähköpostin välityksellä.
- Jani arvioi, että yhteen tapaamiseen olisi syytä varata vähintään 1 tunti.
- Jani on saanut järjestelmän tunnukset vastuuhenkilöltä, joka on myös kehottanut vaihtamaan salasanan heti ensimmäisellä kirjautumis kerralla.
- Jani sopii tapaamisten pito paikaksi aina oman huoneensa, koska tällöin hän voi unohduksen sattuessa jatkaa omia töitään. Tapaaminen pidetään yleensä kuitenkin jossain muussa huoneessa.
- vastuuhenkilön lähettämälista sähköpostissa on nimi, sähköpostiosoite.
    - Matti Kuoppamäki
    - Jaakko Virtanen
    - Jussi Kangasharju
    - Päivi Jukkasalmi-Koskelo
    - Juha Kärkkäinen
    - Mika Kungen
    - Timo Aaltc
    - Petteri Palcjoki
    - Samuel Ojala
    - Tomi Kokkola
    - Paula Evilä
    - Teemu Hanhikoski
    - Kati Aho
    - Jani Kaivoksela
    - Marju Pescnen
- Ensimmäinen ryhmätapaaminen on perjantain 12.10.2007 klo. 10-12 huokassa C222.

- Petterille sopivat ajat:
  - to 25.10 12-13
  - pe 26.10 8-10
  - ma 29.10 9-12

Toteuma:

- Tapaamiseen saapui 10 oppilasta
- Poissa ovat (Marju, Jani, Kati, Teemu, Paula)
- 7 oppilaan kanssa henkilökohtainen tapaaminen saadaan sovittua ryhmätapaamisen aikana. Sovitut ajat ovat:
  - Matti Kuoppamäki pe 26.10 klo 9-10
  - Jaakko Virtanen ma 29.10 klo 10-11
  - Jussi Kangasharju to 25.10 klo 12-13
  - Päivi Jukkasalmi-Koskelo ti 30.10 klo 13-14
  - Juha Kärkkäinen to 1.11 klo 10-11
  - Mika Kungen to 1.11 klo 15-16
  - Timo Aalto to 1.11 klo 13-14

- Petteri, Samuel ja Tomi eivät tiedä aikataulujaan, joten sopiminen jää myöhempään
- Petteri sopi tapaamisen järjestelmän kautta 17.10
- Samuel, Tomi ja Paula varaavat myös aikansa ennen 19.10.ta
  - Samuel pe 26.10 16-17
  - Tomi pe 2.11 16-17
  - Paula ti 30.10 10:30-11:30
- Jani menee 19.10 tarkistamaan kuinka moni on varannut ajat

Kuva 6. Tavoitepohjainen käyttötapaus 1.

KT 2. Henkilökohtaisen tapaamisen varaaminen

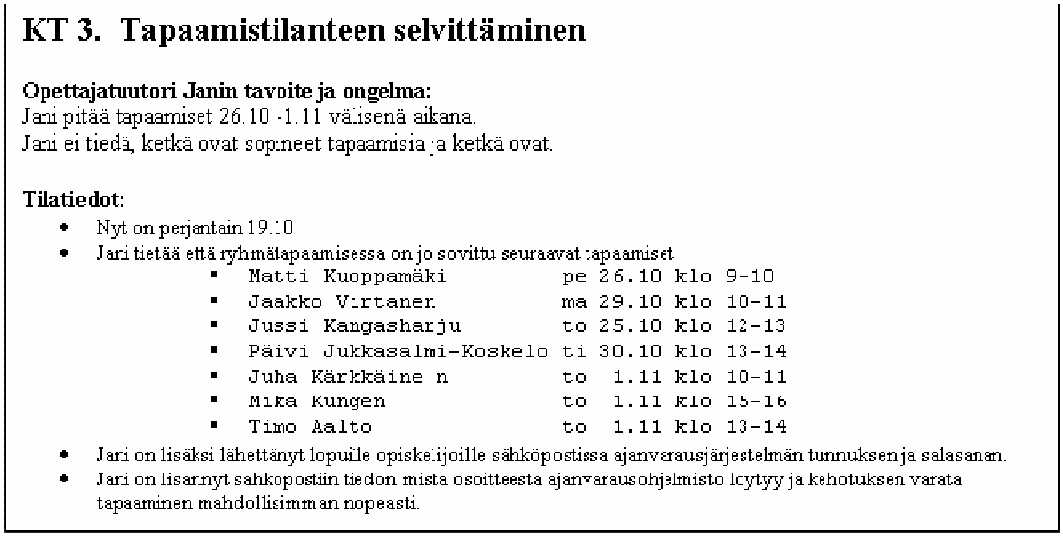
Opiskelija Petterin tavoite ja ongelma:
Petteri on saanut opettajatuutorilta sähköpostilla kehotuksen varata henkilökohtainen tapaaminen ja ohjeet, miten tapaaminen kannattaisi sopia. Petteri tietää, että tapaaminen on sovittava viimeistään 22.10, koska opettajatuutori on mainninut takarajan sähköpostissa.
Petteri tietää, että kurssille kuuluu lukukausittaiset pakolliset henkilökohtaiset tapaamiset ja että hänen olisi sovittava se mahdollisimman pian, jotta tapaamisen sopiminen ei unohdu. Petteri ei ollut ryhmätapaamisessa, joten hänellä ei ole ollut ennen opettajan lähettämää sähköpostia selvyyttä, miten varata tapaaminen.

Tilatiedot:

- Nyt on keskiviikko 17.10
- Sähköpostissa opettaja on kertonut, että kurssilla on ajanvarausjärjestelmä , ja www-osoitteen, mistä ohjelmisto löytyy
- Lisäksi sähköpostissa on Petterin käyttäjätunnus ja salasana ajanvarausjärjestelmään
- Seuraavat tapaamiset on jo sovittu
  - Matti Kuoppamäki pe 26.10 klo 9-10
  - Jaakko Virtanen ma 29.10 klo 10-11
  - Jussi Kangasharju to 25.10 klo 12-13
  - Päivi Jukkasalmi-Koskelo ti 30.10 klo 13-14
  - Juha Kärkkäinen to 1.11 klo 10-11
  - Mika Kungen to 1.11 klo 15-16
  - Timo Aalto to 1.11 klo 13-14

- KT1 löytyy opettajatuutorille sopivat ajat

Kuva 7. Tavoitepohjainen käyttötapaus 2



Kuva 8. Tavoitepohjainen käyttötapaus 3.

6.3 Simulointitestaus

Käyttötapausten pohjalta tehtiin projektien toteuttamille käyttöliittymille simulointitestaus [Laa06d]. Tässä vaiheessa simulointiin, miten Opeapuri-ryhmän ja Opeapu-ryhmän toteuttamilla ohjelmistoilla on mahdollista suorittaa kuvissa 6,7 ja 8 kuvatut tavoitepohjaiset käyttötapaukset. Simuloinnin tuloksena tuotettiin käyttötapauksen läpiviennin kuvaavat kuvasarjat [Han07a, Han07b] molemmista järjestelmistä. Simulointitestauksessa simuloitiin vain kolme käyttötapaus ja edes niiden variaatioita ei työssä simuloitu. Luultavaa on, että käyttötapauksen variaatiot ja useampien käyttötapausten simulointi paljastaisi vielä ison joukon uusia ongelmia.

Simuloinnin ensimmäisessä vaiheessa on tarkoitus selvittää käyttötilanteen paras loppuratkaisu ilman järjestelmän mahdollisesti asettamia rajoituksia [Laa06d]. Tässä tapauksessa se on saada sovittua opiskelijoiden kanssa lukukauden henkilökohtaiset tapaamiset ja niin, että ne ovat mahdollisimman sopivat molemmille osapuolille. Seuraavassa vaiheessa etsittiin järjestelmän tarjoama paras polku käyttötapauksen läpiviemiseen. Polun löydyttyä muodostettiin sen kulusta kuvasarjat, joista käy ilmi, mitä käyttäjä missäkin tilanteessa tekee järjestelmällä. Polusta muodostettiin näyttökuvat siihen asti kunnes suurin osa tapaamisista on saatu sovittua. Tämän jälkeen polusta kirjattiin muistiin havaitut puutteet toiminnoissa ja tietosisällöissä sekä havaitut käytettävyysongelmat.

6.4 Simulointitestauksessa havaitut käyttöliittymäongelmat

Tässä luvussa esitellään ryhmien käyttöliittymistä simulointitestauksen aikana löydetyt käyttöliittymäongelmat. Erityisesti keskitytään keskeisimpiin ja vakavimpiin ongelmiin. Ongelmat on pyritty kuvaamaan käyttötapauksen etenemisjärjestyksessä, jotta ongelmien hahmottaminen olisi mahdollisimman helppoa. Simuloinnin aikana havaittiin, että parhaaksi valitussa toimintastrategiassa on seuraavat vaiheet:

- Ryhmien ja opiskelijoiden luonti.
- Henkilökohtaisten tapaamisten merkitseminen.
- Ryhmätapaamisen aikana sovittavien henkilökohtaisten tapaamisten. merkitseminen.
- Opiskelija varaa itse henkilökohtaisen tapaamisen.
- Opettajatuutori tarkistaa varaustilanteen.

Tuloksena saadut ongelmat on jaoteltu käyttöliittymät-kurssin luentomonisteessa esitetyn jaottelun mukaan seuraaviin kategorioihin [Laa06d]:

- Käyttötapauksen läpimenon estävät ongelmat toiminnoissa ja tietosisällössä
- Tehokkuusongelmat
    - turha mekaaninen työ
    - turha mentaalinen työ
- Opittavuusongelmat

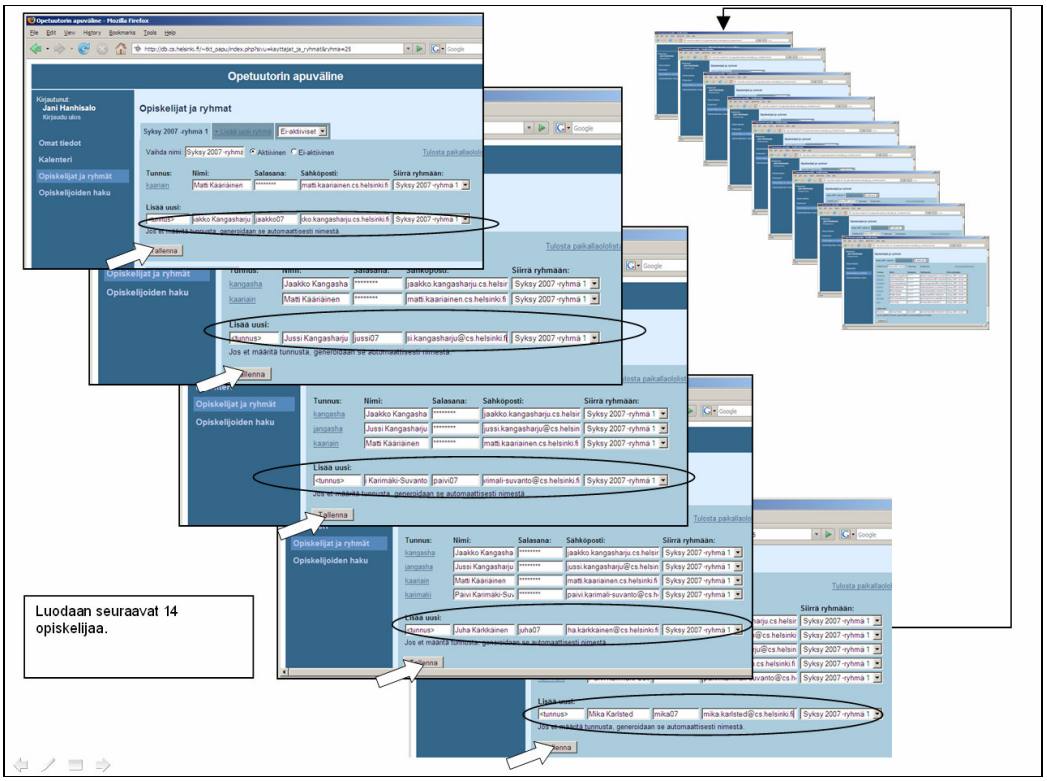
6.4.1 Ryhmien ja opiskelijoiden luonti

Käyttötapauksen ensimmäisessä vaiheessa opettajatuutori luo järjestelmään ryhmän ja opiskelijat, jotta järjestelmään on mahdollista varata henkilökohtaisia tapaamisia ryhmän opiskelijoille.

Ensimmäinen vakava tehokkuusongelma Opeapuri-ryhmän käyttöliittymässä on opiskelijoiden syöttämisessä järjestelmään. Jokainen opiskelija on luotava erikseen toimintoketjulla, jossa ensimmäiseksi opettajatuutori täyttää opiskelijan tiedot kohta kohdalta ja painaa Tallenna-painiketta. Sama ketju toistuu jokaisen opiskelijan kohdalla. Kuvassa 9 on kohta, jossa ongelma on havaittu simulointitestauksessa.

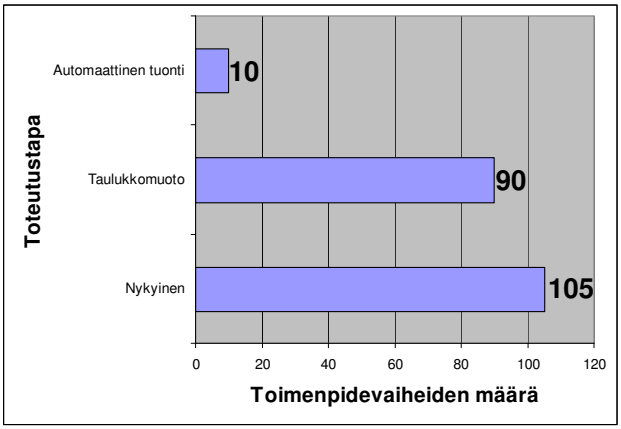
Suoraviivaisin ratkaisu ongelmaan on luoda mahdollisuus syöttää opiskelija suoraan tekstitiedostosta tai mahdollisesti suoraan Kurki-järjestelmästä, jossa ryhmäjako on jo olemassa. Mikäli tämä ei ole mahdollista, voidaan ainakin turhat Tallenna-painikkeen

painallukset poistaa luomalla näytölle valmiiksi esimerkiksi 20 tyhjää riviä, joihin tiedot voidaan täyttää.



Kuva 9. Opiskelijan luonti Opeapuri-ryhmän käyttöliittymässä.

Kuvassa 10 on kuvattu työmäärien erot eri toteutustavoissa. Nykyisellä tavalla toteutettuna viidentoista opiskelijan luonti vaatii pahimmassa tapauksessa 105 toimenpidevaihetta, kun vaiheeksi lasketaan tiedon täyttö, siirtyminen seuraavaan syötekenttään ja Tallenna-painikkeen painaminen. Taulukkomuotoisessa näkymässä luonti vaatisi 90 toimenpidettä. Tekstitiedostovaihtoehdossa tuomisessa, mikäli tiedosto tulisi suoraan vastuuhenkilöltä, valittaisiin vain tiedosto, jossa opiskelijat ovat, ja painettaisiin opiskelijoiden tuomisen laukaisevaa painiketta. Voidaan arvioida, että tämä vaatisi noin 10 toimintoa mukaan lukien tiedoston etsimisen tiedostohallinnasta. Suora tuominen Kurki-järjestelmästä riippuu tietenkin toteutuksesta, mutta voidaan arvioida, että siinä valittaisiin korkeintaan lukukausi, jonka ryhmät haetaan. Tämän jälkeen painettaisiin painiketta, joka hakisi opiskelijoiden tiedot. Menemättä sen enempää ratkaisun yksityiskohtiin, voidaan arvioida, että siinäkin olisi korkeintaan 10 toimintovaihetta. Näin ollen tehokkuusongelman voidaan arvioida olevan vakava. Huomioitavaa on, että taulukkomuotokaan ei vähennä vaadittavia toimenpidevaiheita kuin 14 prosenttia, koska edelleen jokaisen oppilaan tiedot on syötettävä yksitellen järjestelmään.



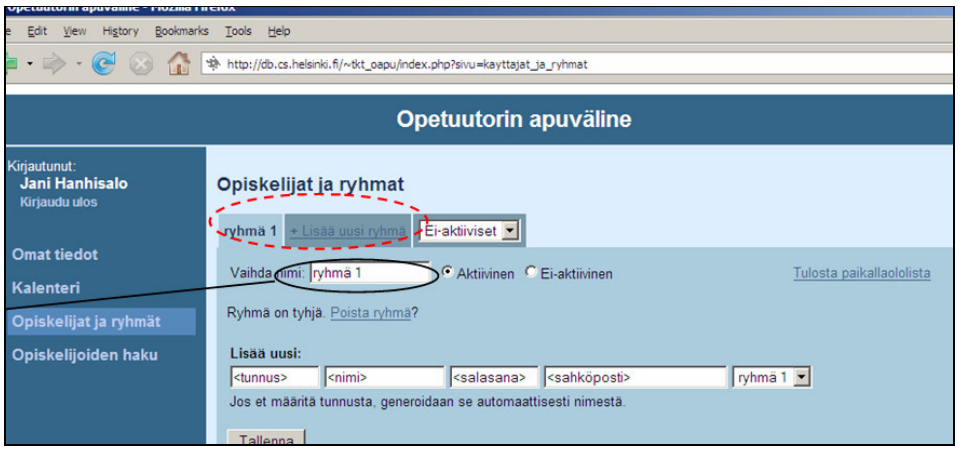
Kuva 10. Toteutustapojen vaatimien toimenpidevaiheiden määrät.

Voidaan todeta, että jos järjestelmään olisi suunniteltu oppilaiden massasyöttö, olisi ongelmalta suurella todennäköisyydellä vältytty. Huomioitavaa on, että Opeapuri-ryhmä on kyllä kirjannut vaatimuksiin massasyötön, mutta sen prioriteetti on 3 (pyritään toteuttamaan). Toimintoa ei ole kuitenkaan toteutettu. Vaatimusmäärittelyvaiheessa ei siis ole ollut täyttä ymmärrystä siitä, kuinka oleellinen toiminto massasyöttö tulee olemaan, ja näin ollen sitä ei ole toteutettu.

Simuloitaessa oppilaiden syöttämistä järjestelmään havaittiin muutama muukin pienempi tehokkuusongelma. Esimerkiksi käyttäjän syötettyä opiskelijan nimen ja siirryttyään seuraavaan ruutuun ja tyhjennettyään sen. Käyttäjän tarkistaessa, kirjoittiko hän nimen varmasti oikein katsomalla sen saamastaan opiskelijalistasta tai huomion kiinnittyessä hetkeksi johonkin muuhun kuvan 11 mukaisesti, seuraavan kentän otsikko ei ole enää näkyvissä. Näin käyttäjä joutuu päättelemään luotujen opiskelijoiden tiedoista, mitä seuraavaan kenttään pitikään syöttää. Ensimmäisen opiskelijan luonnissa käyttäjä ei voi tarkistaa otsikkoa mistään, joten käyttäjä joutuu pitämään otsikon mielessään, mikä on turhaa mentaalista työtä. Huomioitavaa on, että ainakin otsikoiden sijoitteluongelma poistuu jo muuttamalla tiedot taulukkomuotoon.

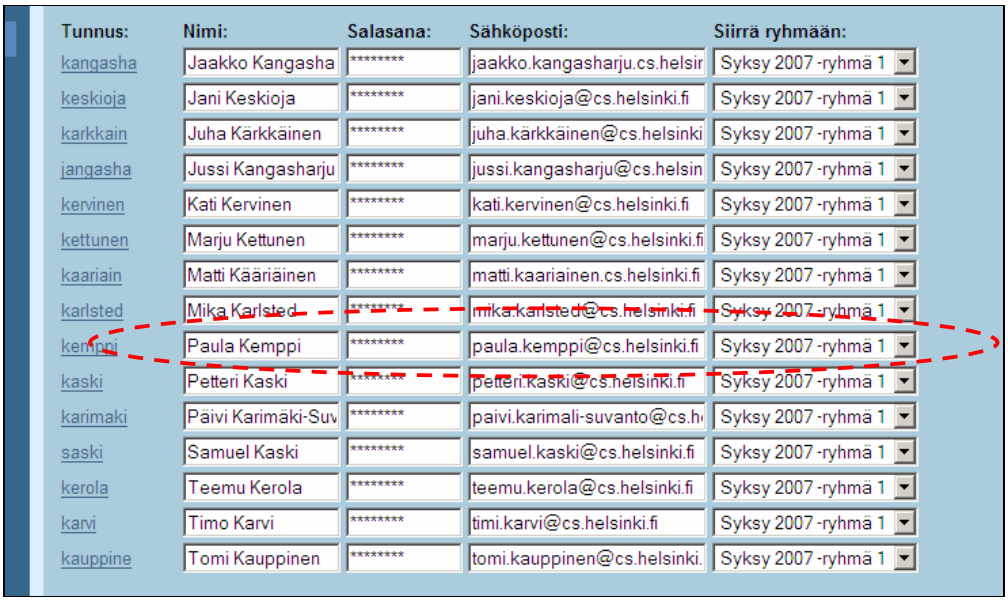
Kuva 11. Tehokkuusongelmia Opeapuri-ryhmän käyttöliittymässä.

Ryhmien ja opiskelijoiden luonnissa havaittiin lisäksi opittavuusongelma. Opettajatuutorin luodessa ensimmäistä ryhmää järjestelmä on luonut valmiiksi yhden ryhmän ja antanut sille nimeksi ”Ryhmä 1” (kuva 12). Vieressä on kuitenkin ”Lisää uusi ryhmä” -linkki, jota käyttäjän houkuttaa painaa, koska hän on luomassa uutta ryhmää. Linkin painaminen luo kuitenkin toisen ryhmän. Käyttäjää houkuttaa painaa painiketta, vaikka sen painaminen käyttötilanteessa virheelliseen toisen ryhmän 2 luontiin. Käyttäjän on ymmärrettävä, että ryhmä on jo luotu valmiiksi eikä hänen tarvitse luoda sitä uudestaan.



Kuva 12. Opittavuusongelma Opeapuri-ryhmän ryhmien luonnissa.

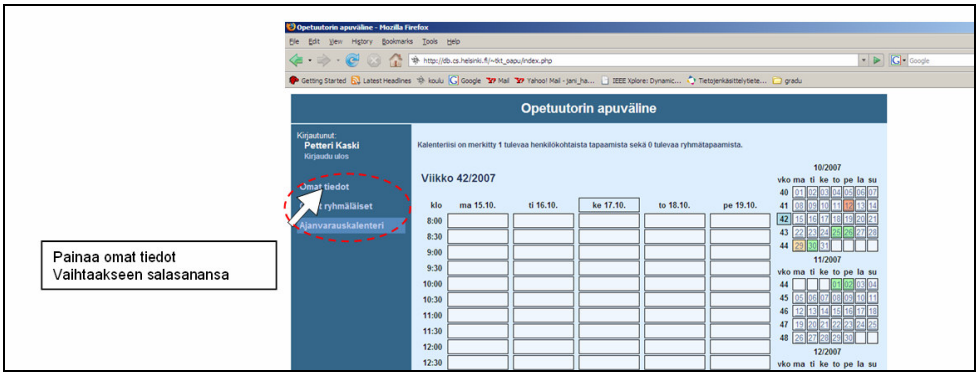
Simuloinnissa havaittiin opittavuusongelmaksi myös se, että käyttäjän täytettyä esimerkiksi Paula Kempin tiedot ja painettuaan Tallenna-painiketta luotu opiskelija ilmestyy opiskelijalistaan, kuvan 13 mukaisesti keskelle, ja rivi näyttää identtiseltä muihin riveihin verrattuna. Opittavuusongelmaksi tilanteen tekee se, että käyttäjän on vaikea hahmottaa, onnistuiko luonti, koska uuden rivin löytäminen listasta on työlästä. Ongelma voitaisiin poistaa korostamalla luotu rivi, jolloin käyttäjä saisi palautteen lisäyksen onnistumisesta.



Kuva 13. Luodun opiskelijan näkyminen Opeapuri-ryhmän käyttöliittymässä.

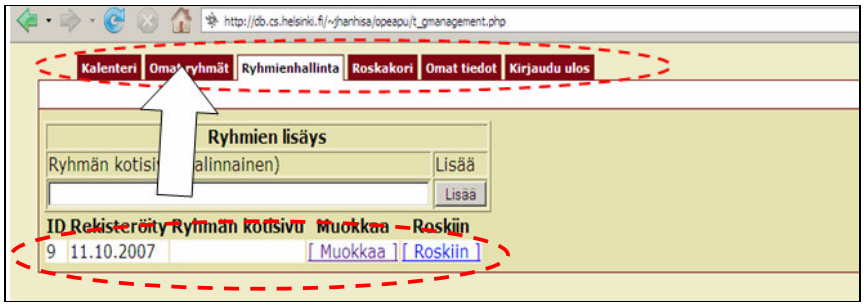
Mielenkiintoinen havainto on lisäksi se, että Opeapuri-ryhmän käyttöliittymässä on virheellisten tietojen käsittelyssä myös sellainen vakava opittavuus- ja tehokkuus-ongelma, jota simulointitestauksella ei havaita. Mikäli käyttäjä syöttää opiskelijan lisäyskenttiin virheellistä tietoa tai jättää syöttämättä osan tiedoista ja painaa tämän jälkeen Tallenna-painiketta, näytölle ilmestyy luontisivu ilman täytettyjä tietoja. Opiskelijaa ei luotu, eikä epäonnistumisesta tule mitään informaatiota. Simulointitestaus ei kuitenkaan havaitse tätä ongelmaa, koska siinä simuloidaan vain käyttötapauksen läpiviemisen oikeaa polkua.

Opeapuri-ryhmän käyttöliittymässä on opittavuusongelma myös kirjauduttaessa sisään ensimmäistä kertaa, kun käyttäjän tulisi vaihtaa salasana. Käyttäjän kirjautuessa järjestelmään avautuu kalenterisivu (kuva 14). Käyttäjän on seuraavaksi painettava Omat tiedot -linkkiä päästäkseen vaihtamaan salasanansa. Näin käyttäjän on siis itse ymmärrettävä mennä vaihtamaan salasana Omat tiedot -sivulta. Opeapu-ryhmän käyttöliittymässä ongelma on poistettu siten, että käyttäjä kirjautuessa ensimmäistä kertaa järjestelmään hänelle aukeaa suoraan Omat tiedot -sivu, ja käyttäjälle annetaan vihje, että salasana on syytä vaihtaa, koska sitä ei ole muutettu. Näin järjestelmä tukee salasanan vaihdon muistamista.



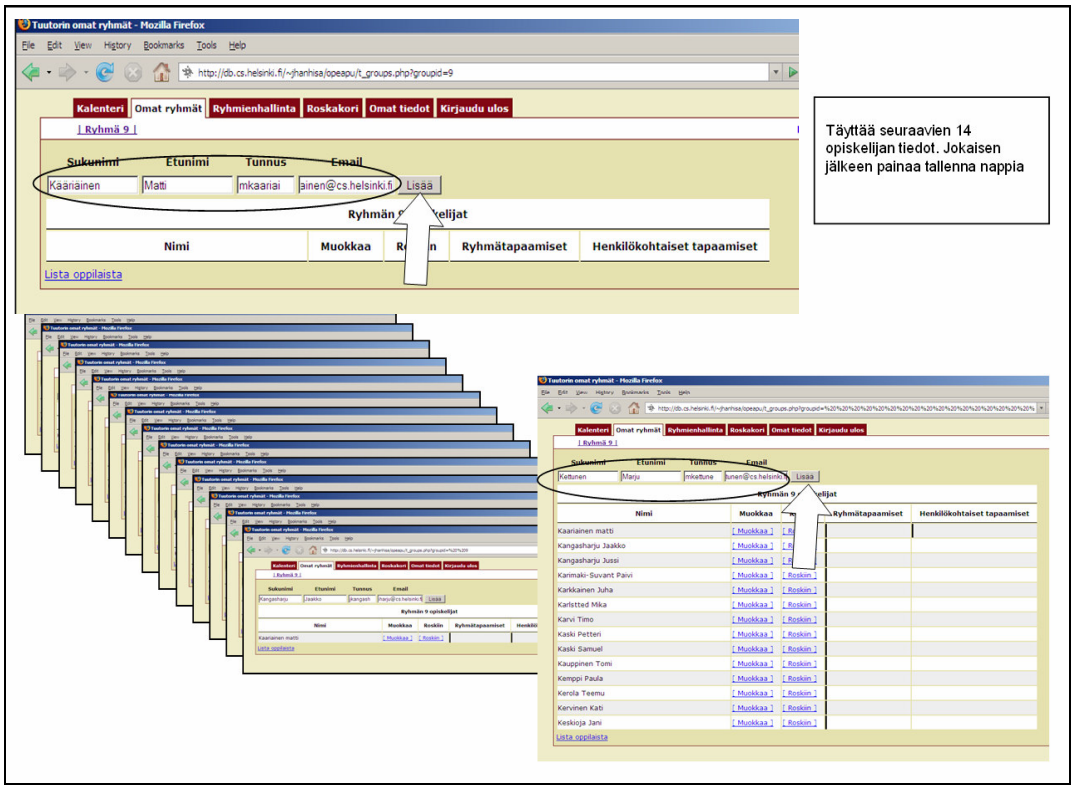
Kuva 14. Opittavuusongelma Opeapuri-ryhmän käyttöliittymässä.

Opeapu-ryhmän käyttöliittymässä ensimmäinen ongelma ryhmien luonnissa on opittavuusongelma valikossa (kuva 15). Käyttäjän luodessa uutta ryhmää hän valitsee Ryhmienhallinta-välilehden. Valikossa on lisäksi Omat ryhmät -välilehti. Käyttäjälle ei ole selvää, kumpi hänen tulee valita luodessaan uutta ryhmää, koska molemmissa viitataan ryhmiin. Todellisuudessa Ryhmienhallinta-välilehti on ryhmien luontia ja poistoa varten ja Omat ryhmät -välilehdellä lisätään luotuun ryhmään opiskelijoita. Luotuaan ryhmän käyttäjä painaa Omat ryhmät -välilehteä, jotta hän pääsee lisäämään luotuun ryhmään opiskelijoita. Jaottelusta seuraa nyt tehokkuusongelma, koska käyttäjä joutuu ryhmän luotuaan siirtymään turhaan toiselle välilehdelle, jotta voi lisätä opiskelijat ryhmään (kuva 15).



Kuva 15. Turhaa navigointia Opeapu-ryhmän käyttöliittymässä.

Opiskelijoiden luonnissa Opeapu-ryhmällä on sama tehokkuusongelma (kuva 16) kuin Opeapuri-ryhmällä. Käyttäjä joutuu luomaan opiskelijat yksi kerrallaan, mistä seuraa turhaa mekaanista työtä. Toiminnon suorittaminen Opeapu-ryhmän käyttöliittymässä vaatii samat 105 toimenpidevaihetta kuin toiminnon toteutus Opeapuri-ryhmän käyttöliittymässä. Ongelman ollessa sama myös ratkaisut ovat samat, joten niitä ei käsitellä enää uudelleen.



Kuva 16. Tehokkuusongelma opiskelijoiden lisäämisessä Opeapu-ryhmän käyttöliittymässä.

Lisäksi käyttäjän luonnissa on muutama pienempi käytettävyysongelma. Ensimmäinen opittavuusongelma havaitaan, kun käyttäjä syöttää käyttäjätunnusta. Käyttäjätunnuksen maksimipituus on kahdeksan merkkiä. Tätä ei kuitenkaan mainita missään, ja käyttäjä huomaa asian vasta kirjoittaessaan kenttään yhdeksännen merkin, joka ei ilmestykään ruutuun. Tehokkuusongelma havaitaan käyttäjän syöttäessä sähköpostiosoitetta. Ongelma on sähköpostikentän lyhyys, jonka seurauksena koko sähköpostiosoite ei näy kerralla ruudulla. Näin sen tarkistaminen vaatii turhia toimenpiteitä.

Vakava tehokkuusongelma havaitaan käyttäjän syöttäessä sukunimi-kenttään opiskelijan nimeä ”Karimäki-Suvanto”. Sukunimi-kentän pituus on rajattu 15 merkkiin. Rajauksen johdosta kyseistä sukunimeä ei voida tallentaa kokonaisena, vaan vain 15 ensimmäistä merkkiä voidaan tallentaa järjestelmään. Tämä ongelma lienee enemminkin ohjelmointivirhe kuin käytettävyysongelma.

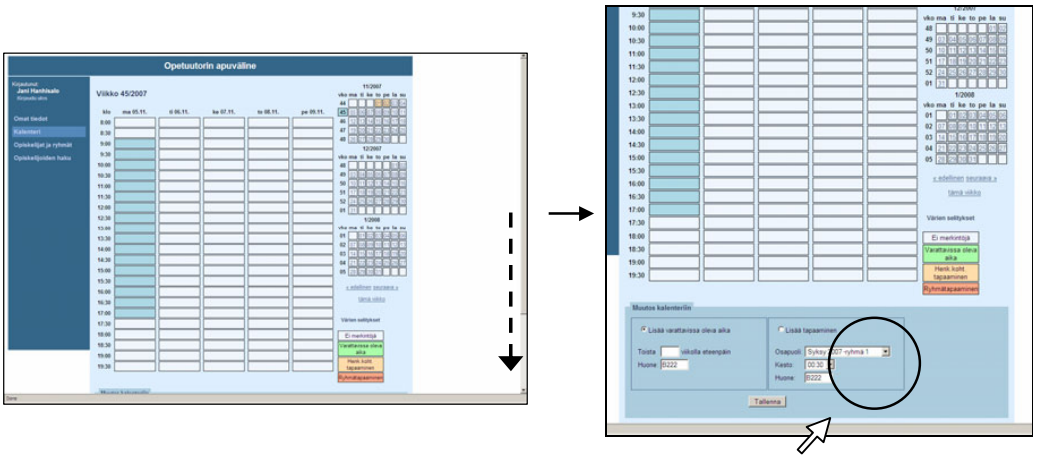
Molempien ryhmien käyttöliittymässä ryhmien ja opiskelijoiden luontitoiminnot on rakennettu siten, että opettajatuutori joutuu luomaan opiskelijat yksi kerrallaan, mikä aiheuttaa vakavan tehokkuusongelman. Kumpikaan käyttöliittymä ei tarjoa mahdollisuutta tuoda opiskelijoita esimerkiksi Kurki-järjestelmästä tai tekstitiedostosta.

Keskeisten toimintojen puuttumisen lisäksi molemmissa käyttöliittymissä on havaittavissa hyvin samankaltaisia huolimattomuusvirheitä. Näitä ovat esimerkiksi liian lyhyet kentät ja vääränlaiset kenttien rajoitukset, jotka johtavat pieniin tehokkuus- ja opittavuusongelmiin.

6.4.2 Henkilökohtaisille tapaamisille merkittävät ajat

Ryhmän ja opiskelijoiden luonnin jälkeen opettaja merkitsee järjestelmän kalenteriin ajat, jolloin hän haluaa järjestää henkilökohtaiset tapaamiset. Merkitsemisen jälkeen opiskelijoiden on mahdollista varata opettajan merkitsemiä aikoja.

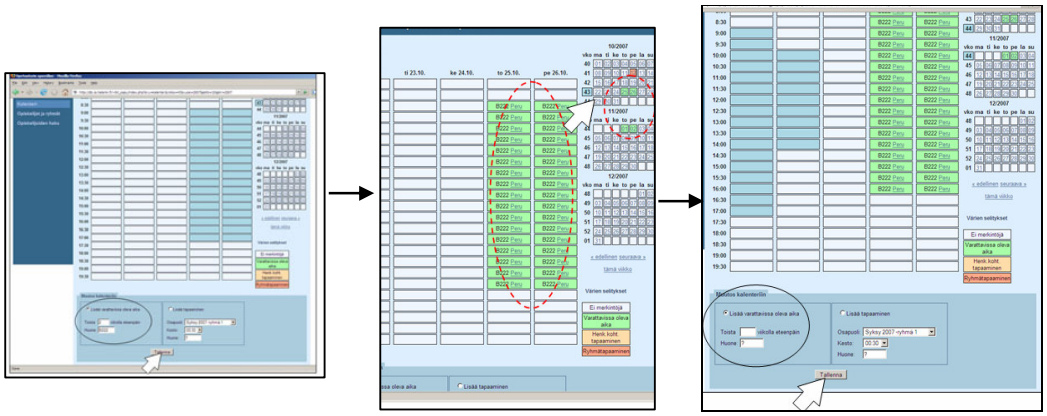
Molemmat ryhmät ovat toteuttaneet aikojen valitsemisen kalenterin kautta. Kalenterin toteutukset eroavat kuitenkin hyvin paljon toisistaan. Opeapuri-ryhmän ensimmäinen tehokkuusongelma aikojen merkitsemisessä on sivun jatkuva vieritys ylös ja alas (kuva 17). Ongelma esiintyy useissa kohdissa käyttötapauksen suoritusta. Esimerkiksi silloin, kun opettajatuutori merkitsee aluksi kalenterista henkilökohtaisille tapaamisille sopivat ajankohdat, ja vierittää sivua alaspäin syöttääkseen aikoihin liittyvät tiedot ja painaakseen Tallenna-painiketta. Ongelma voitaisiin ratkaista pienentämällä kalenterin laatikoita ja tiivistämällä ruudun ylälaidassa olevia otsikoita.



Kuva 17. Turha vieritys Opeapuri-ryhmän käyttöliittymässä.

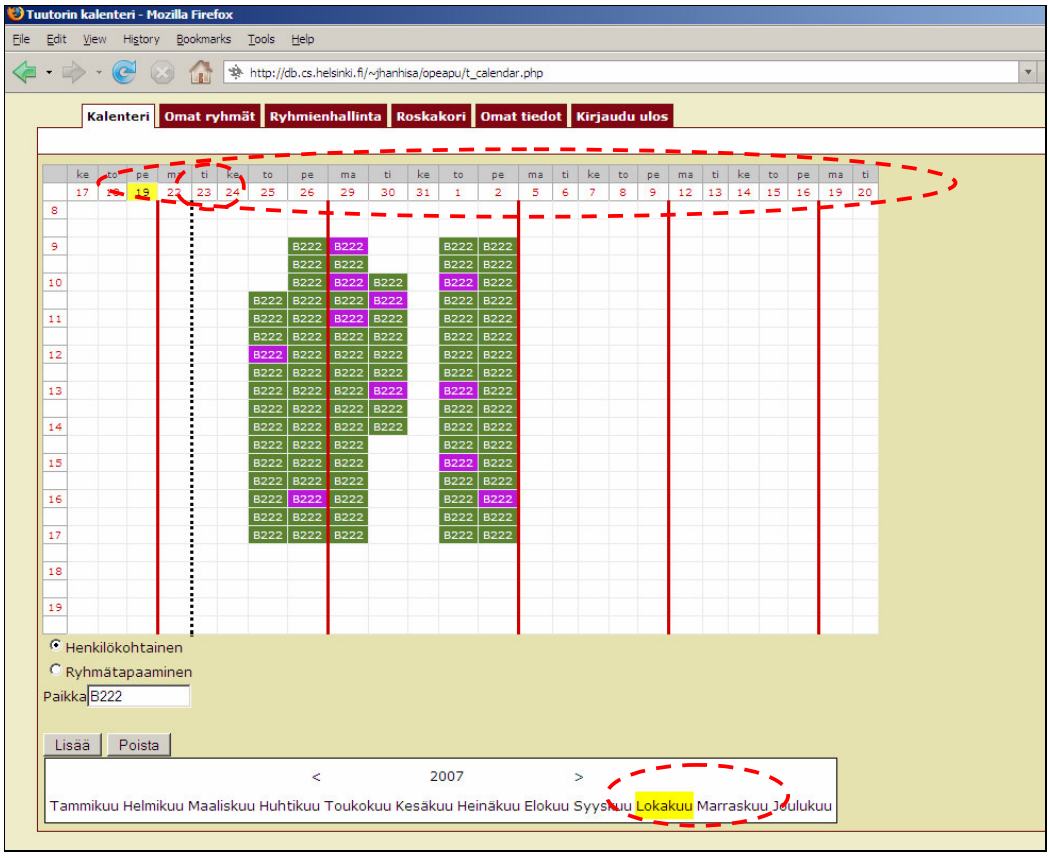
Turhaa työtä aiheuttava ongelma havaittiin kuvasarjan siinä kohdassa, jossa opettaja merkitsee henkilökohtaisille tapaamisille sopivia aikoja. Opettajalle sopivat ajat jakautuvat kahdelle eri viikolle, joten opettaja joutuu merkitsemään ajat kahdessa osassa. Aluksi opettaja syöttää kuvan 18 mukaisesti ensimmäisen viikon tunnit ja tarkistaa lomakkeen tiedot ja painaa Tallenna-painiketta. Tämän jälkeen opettaja vaihtaa oikealla olevasta kalenterista seuraavan viikon näkyviin. Sen jälkeen opettaja täyttää seuraavalla viikolla hänelle sopivat ajat, vierittää sivun alas, tarkistaa tiedot ja painaa

Tallenna-painiketta. Ongelma johtuu kalenterin rajaamisesta siten, että kerralla näkyy ainoastaan viikko.



Kuva 18. Turhaa siirtymistä viikosta toiseen Opeapuri-ryhmän käyttöliittymässä.

Opeapu-ryhmän kalenterista henkilökohtaisille tapaamisille sopivien aikojen merkitseminen on hyvin suoraviivaista. Opettaja pystyy kerralla valitsemaan kaikki haluamansa ajat ja painamaan sen jälkeen Lisää-painiketta, jonka jälkeen opiskelijoiden on mahdollista varata merkittyjä aikoja. Kalenterin keskeisin ongelma on sen ulkoasu, joka johtaa tehokkuus- ja opittavuusongelmiin (kuva 19). Kun opettajatuutori syöttää henkilökohtaisille tapaamisille sopivia aikoja, ovat ensimmäiset syötettävät ajat 25. lokakuuta. Opettaja ei kuitenkaan näe kalenterin yläpuoleisesta päiväjanasta, mille kuukaudelle mikäkin päivä kuuluu. Todellisuudessa lokakuun päivät alkavat vasemmasta laidasta, ja kalenterin loppuosan päivät ovat marraskuuta. Tämän käyttäjä voi päätellä ainoastaan siitä, että alalaidassa on lokakuu merkittynä keltaisella, ja päiväjanalla 19. päivä, joka oli nykyinen päivä, on myös keltainen. Käyttäjän on opittava ymmärtämään, että alalaidasta valittu kuukausi alkaa aina vasemmasta laidasta.



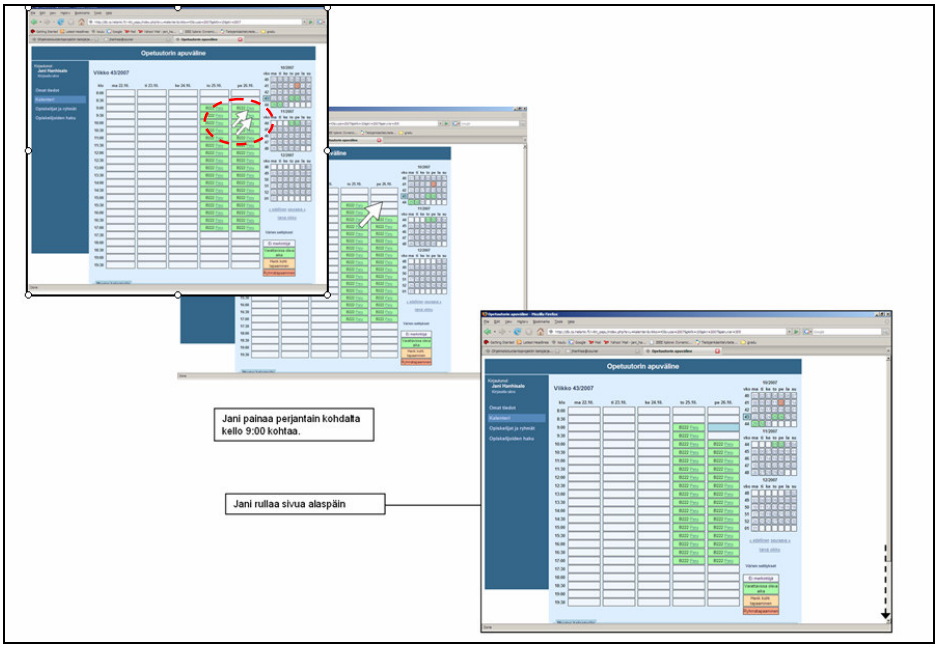
Kuva 19. Opeapu-ryhmän kalenterisivun ulkoasu.

Kaiken kaikkiaan Opeapuri-ryhmän kalenterinäkymä on selvästi valmiimman näköinen. Sen suurin heikkous on turha navigointi eri viikkojen välillä, koska kalenterissa näkyy kerralla ainoastaan viikko. Opeapuri-ryhmän ratkaisussa on kerralla nähtävissä yli kuukauden jakso, jolloin viikolta toiselle siirtymistä ei tarvita. Toisaalta kalenterinäkymän suurin heikkous on tietojen sijoittelun epäloogisuus.. Vaikuttaa kuitenkin vahvasti siltä, että mikäli Opeapu-ryhmän käyttöliittymällä simuloitaisiin käyttötapauksia, joissa jouduttaisiin navigoimaan kuukaudesta toiseen ilmenisi, myös siinä lisää ongelmia. Tässä työssä simuloitu käyttötapaus ei niitä kuitenkaan esiin tuonut.

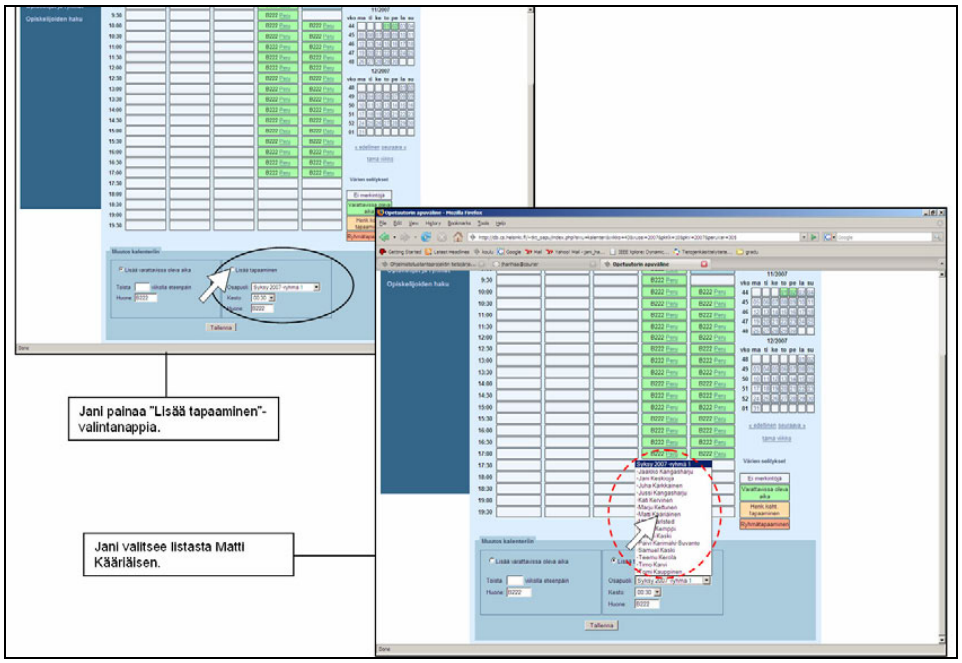
6.4.3 Ryhmätapaaminen

Opeapuri-ryhmän käyttöliittymässä opettajan merkitessä ryhmätapaamisen aikana sovittuja tapaamisia järjestelmään havaittiin useita tehokkuusongelmia. Kuvissa 20 ja 21 on näyttökuvasarjan kohdat, joissa ongelmat ilmenivät. Ongelmat havaittiin, kun opiskelija on ilmoittanut sähköpostilla opettajalle, että hänelle sopisi henkilökohtainen tapaaminen esimerkiksi perjantaina 26.10. klo 9-10. Tällöin opettaja painaa kalenterista ajankohdan laatikoiden Peruuta-linkkejä. Opettaja valitsee kalenterista aloitusajaksi

perjantain kello 9:00 kohdalla olevan laatikon painamalla sitä. Tämän jälkeen opettaja vierittää sivun alas ja valitsee Lisää tapaaminen -valintapainikkeen. Hän painaa Osapuoli-pudotusvalikon auki ja valitsee listasta opiskelijan nimen. Tämän jälkeen opettaja avaa Kesto-pudotusvalikon, valitsee siitä kestoksi yhden tunnin ja painaa Tallenna-painiketta.



Kuva 20. Tehokkuusongelma Opeapuri-ryhmän käyttöliittymässä. Käyttäjä joutuu rullaamaan näyttöä jatkuvasti.



Kuva 21. Opettajatuutori merkitsee henkilökohtaisen tapaamisen ryhmätapaamisen aikana Opeapuri-ryhmän käyttöliittymässä.

Ensimmäinen henkilökohtaisten tapaamisten merkitsemisessä havaittu tehokkuus-ongelma Opeapuri-ryhmän käyttöliittymässä on, että merkitäkseen kalenteriin henkilökohtaisen tapaamisen opettajatuutorin on poistettava ensin aiemmin merkitsemänsä varattavissa olevat ajat, ja vasta sen jälkeen hän voi merkitä siihen henkilökohtaisen tapaamisen (kuva 20). Ongelma voitaisiin poistaa sallimalla opettajatuutorin valita myös vihreä laatikko varatessaan henkilökohtaisia tapaamisia. Tällöin opettajan ei tarvitsisi poistaa aluksi tapaamisille sopivia aikoja ennen varauksen tekoa, vaan hän voisi merkitä opiskelijan varauksen suoraan kalenteriin.

Opittavuusongelmia on lisäksi alalaidan syötekentissä, joihin merkinnän tiedot täytetään (kuva 21). Ongelmat havaittiin simuloinnissa, kun opettaja täyttää henkilökohtaiseen tapaamiseen liittyviä tietoja. Tällöin käyttäjän on muistettava joka kerta valita Lisää tapaaminen -valintapainiketta. Käyttäjän on myös tiedettävä, että samassa Osapuoli-pudotusvalikossa on ryhmät ja opiskelijat. Huomioitavaa on, että valintapainikkeen ongelma paljastui vasta tässä vaiheessa, eikä merkittäessä vapaana olevia aikoja. Syy tähän on, että täytettäessä vapaana olevia aikoja ei valintapainikkeeseen tarvinnut koskea, koska opettajan ainut ryhmä oli jo valmiiksi valittuna.

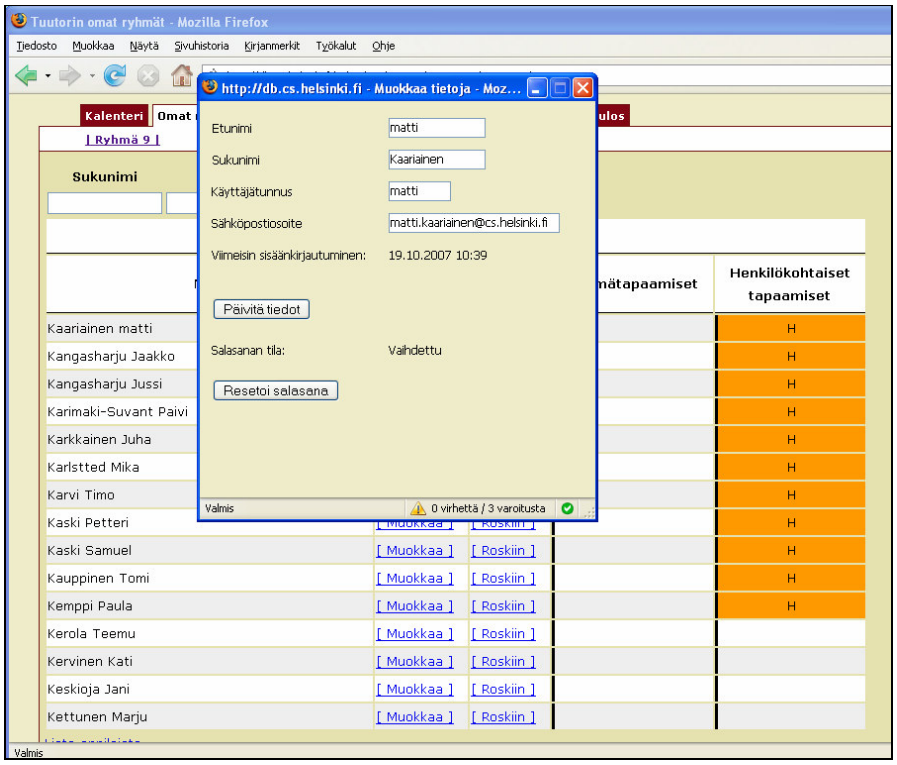
Kun opettajatuutori lisää ryhmätapaamisessa useita henkilökohtaisia tapaamisia peräjälkeen, käyttöliittymässä on tehokkuusongelma. Opettajatuutori joutuu joka kerta valitsemaan Lisää tapaaminen -valintapainikkeen ja vaihtamaan tapaamisen kestoksi

yhden tunnin. Tallenna-painikkeen painamisen jälkeen. Opiskelijan valinnassa havaittiin lisäksi lievä tehokkuusongelma, koska käyttäjä joutuu opiskelijan valitakseen painamaan Osapuoli-pudotusvalikkoa, jotta hän voi valita opiskelijan nimen listasta (kuva 21).

Opiskelijat, jotka eivät olleet tapaamisessa tai eivät vielä ole osanneet sanoa, mikä aika heille sopisi, varaavat ajat omilla tunnuksillaan. Käyttöliittymässä havaittiin puuttuva toiminnallisuus, kun opettajan on ryhmätapaamisen jälkeen lähetettävä opiskelijoille heidän käyttäjätunnuksensa ja salasanansa. Käyttöliittymästä ei ole mahdollista lähettää tunnuksia ja salasanoja opiskelijoille. Tämän seurauksena opettajalle syntyy tarve nähdä luomiensa oppilaiden tunnukset ja salasanat, jotta hän voisi lähettää ne opiskelijoille. Opettajatuutori ei kuitenkaan näe Opeapuri-ryhmän käyttöliittymästä kuin käyttäjien tunnukset. Seurauksena tästä opettajatuutorin on opiskelijoiden luontivaiheessa tallennettava salasana/tunnus-yhdistelmät esimerkiksi tekstitiedostoon tai muistettava jollain muulla keinolla opiskelijoille antamansa salasanat, jotta hän voi lähettää tunnukset ryhmätapaamisen jälkeen. Opeapu-ryhmän järjestelmässä salasanana on käyttäjätunnus, joten siinä salasanaongelmaa ei esiinny.

Tunnusten lähetyksen mahdollistaminen järjestelmän kautta poistaisi ongelman. Nyt opettajatuutori joutuu käyttämään järjestelmän ulkopuolisia välineitä lähettääkseen tunnukset opiskelijoille. Tämä tapahtuisi todennäköisesti sähköpostin välityksellä. Puuttuvaa toiminnallisuutta ei havaittu opiskelijoiden luonnissa, vaan vasta ryhmätapaamisen jälkeen toteutuvissa käyttötilanteissa.

Opeapu-ryhmän käyttöliittymässä opettaja ei myöskään pysty lähettämään opiskelijoiden tunnuksia kootusti, mikä on puuttuva toiminnallisuutta. Järjestelmä antaa onneksi luoduille opiskelijoille salasanaksi syötetyn käyttäjätunnuksen, joten opettajan ei tarvitse kuin kirjata tunnukset ylös. Simuloitaessa tunnusten lähetystä havaittiin tunnusten kerääminen järjestelmästä todella aikaa vieväksi, koska tunnus näkyy ainoastaan opiskelijan tietojen muokkaus sivulla. Näin opettajatuutori joutuu siirtymään pääikkunan ja muokkausikkunan välillä, jotta näkee opiskelijalle antamansa tunnukset (kuva 22). Tästä seuraa tehokkuusongelma, joka voitaisiin korjata asettamalla tunnukset näkyviin opiskelijalistaukseen.



Kuva 22. Tehokkuusongelma tunnusten selvittämisessä Opeapu-ryhmän käyttöliittymässä .

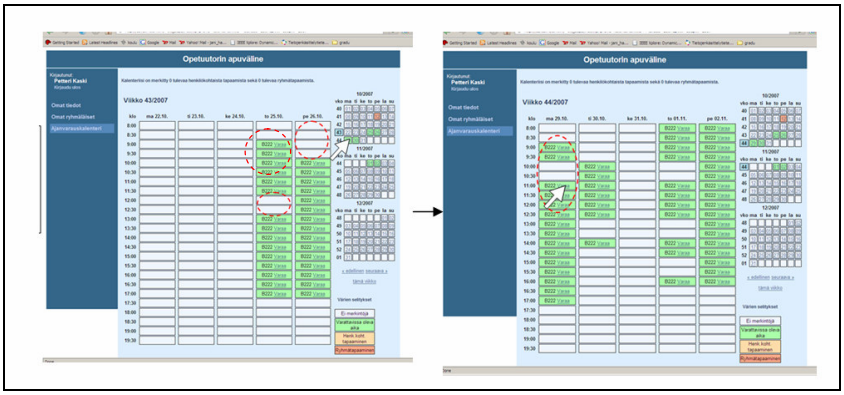
Opeapu-ryhmän käyttöliittymässä havaittiin vakava puute toiminnoissa simuloitaessa ryhmätapaamisen kulkua. Opettajatuutori ei voi merkitä järjestelmän kautta henkilökohtaisia tapaamisia tai muokata jo sovittuja tapaamisia. Käyttöliittymässä ainoastaan opiskelija voi varata henkilökohtaisia tapaamisia. Tämä johtaa siihen, että paras tapa suorittaa käyttötapaus käyttöliittymällä onkin jakaa ryhmätapaamisessa paikalla olijoille tunnukset ja lähettää poissaolijoille sähköpostilla heidän tunnuksensa ja pyytää opiskelijoita varaamaan henkilökohtainen tapaaminen saamillaan tunnuksilla.

6.4.4 Opiskelija varaa henkilökohtaisen tapaamisen

Ryhmätapaamisen jälkeen ryhmän opiskelijat saavat sähköpostilla tunnukset ja ohjeet varaukseen tekoon. Opiskelijat kirjautuvat jossain vaiheessa järjestelmään sisälle ja varaavat itselleen sopivan ajan.

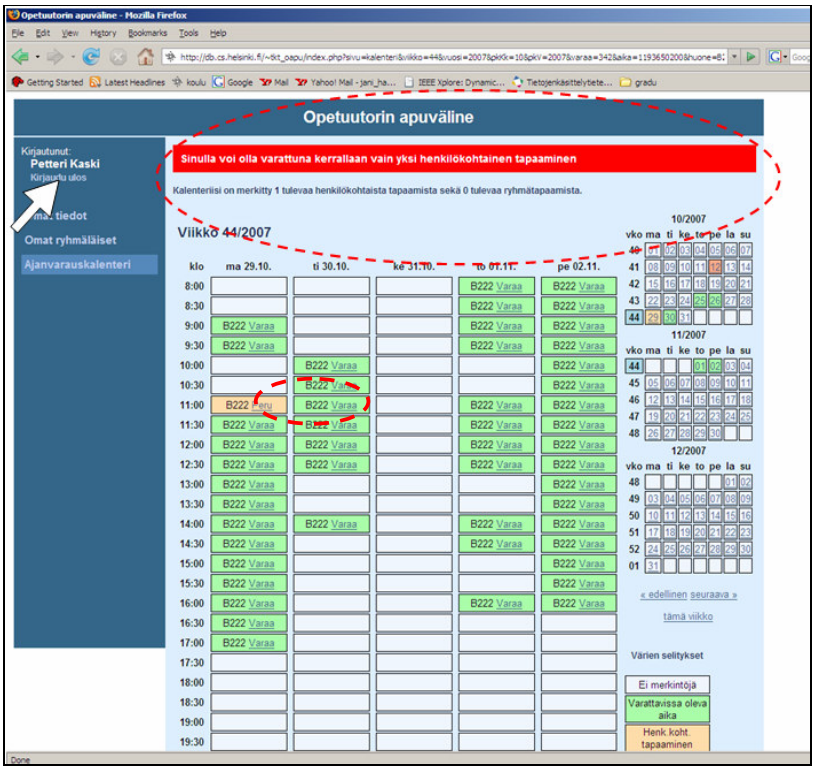
Käyttöliittymässä havaittiin turhaa navigointia ja turhaa mentaalista työtä simuloitaessa, miten opiskelija selvittää henkilökohtaisen tapaamiselle sopivaa aikaa (kuva 23), koska opettajatuutorin merkitsemät vapaat ajat ovat kahdella eri viikolla. Tällöin opiskelija joutuu pitämään toisen viikon vapaat ajat mielessään, kun hän katsoo toista viikkoa, koska kaikki päätöksentekoon vaikuttava tieto ei ole kerralla näkyvissä. Ongelma

havaittiin, kun simuloitiin tilannetta, jossa opiskelijalle sopisi ensimmäiseltä viikolta muutama aika, mutta hän katsoo vielä seuraavan viikon vapaat ajat, koska silloin opiskelija on muutenkin koululla. Tällöin opiskelija painaa kalenterista seuraavan viikon vihreänä olevaa päivää, jolloin esille tulee seuraava viikko. Nyt opiskelijan on kuitenkin pidettävä mielessä edellisen viikon ajat, jotta vertailun tekeminen on mahdollista.



Kuva 23. Turhaa navigointia Opeapuri-ryhmän käyttöliittymässä.

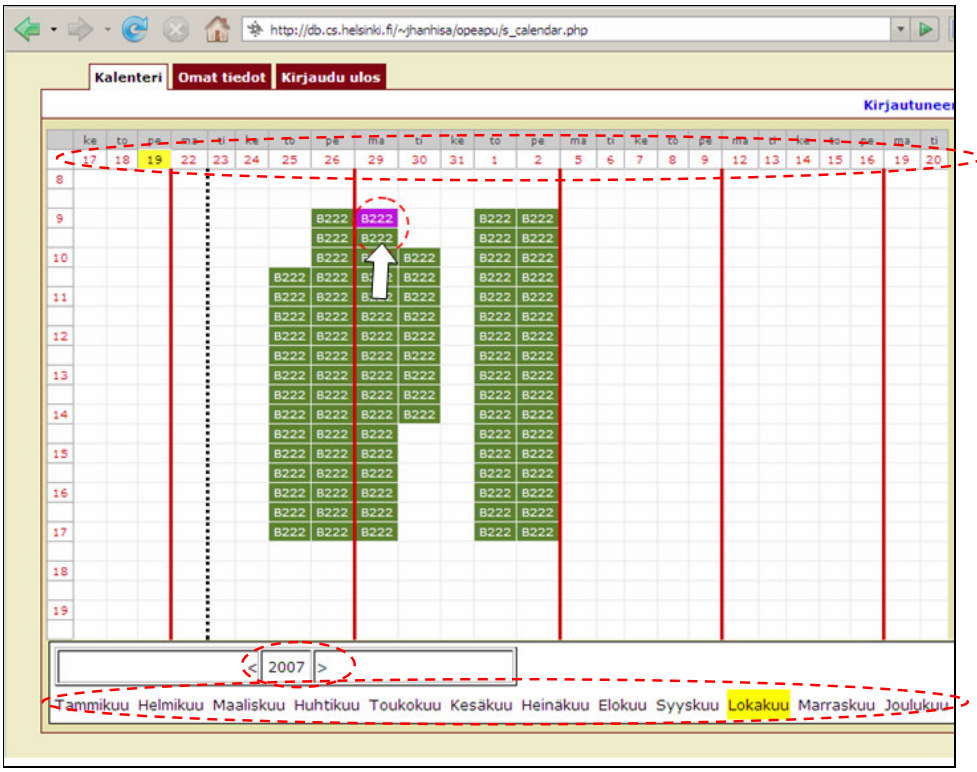
Suurin ongelma käyttöliittymässä on kuitenkin se, että opiskelija voi varata vain yhden 30 minuuttin pituisen henkilökohtaisen tapaamisen (kuva 24). Haastateltava opettaja-tuutori mainitsi, että hän varaa tapaamiselle yleensä yhden tunnin. Kun opiskelija yrittää varata tunnin pituista tapaamista, hän valitsee aluksi kalenterista yhden 30 minuutin laatikon ja varaa sen. Tämän jälkeen opiskelija valitsee toisen laatikon ja painaa Tallenna-painiketta. Nyt järjestelmä ilmoittaa, että opiskelijalla voi olla kerralla vain yksi tapaamisaika varattuna, mikä tarkoittaa siis yhden laatikon varaamista. Näin ollen opiskelijan ei ole mahdollista varata järjestelmästä oikean kokoista aikajaksoa. Vaikuttaa ilmeiseltä, että tapaamisten kestoaika on opettajatuutorikohtainen. Ongelma voitaisiin poistaa tarjoamalla opettajatuutorille mahdollisuus määritellä henkilökohtaiseen tapaamiseen varattavan aikajakson pituus esimerkiksi 15 minuutin tarkkuudella.



Kuva 24. Väärän pituinen varaus Opeapuri-ryhmän käyttöliittymässä.

Opeapu-ryhmän käyttöliittymässä opiskelijan luonnissa on seuraavat käyttöliittymä-ongelmat. Kuten Opeapuri-ryhmänkin käyttöliittymässä, myös Opeapu-ryhmän käyttöliittymässä varauksen pituus on rajattu 30 minuuttiin. Käyttöliittymässä ei ole turhaa navigointia, vaan kaikki ajat näkyvät kerralla. Käyttöliittymässä on kuitenkin muutamia opittavuusongelmia. Opiskelijan varauksessa tulivat esiin samat ongelmat kuin opettajan merkitessä henkilökohtaisille tapaamisille sopivia aikoja. Näitä olivat kuukauden nimen puuttuminen aikajanalta ja kuukausivalinnan sijoittaminen sivun alalaitaan.

Lisäksi käyttöliittymässä havaittiin opittavuusongelma opiskelijan varatessa kalenterista henkilökohtaista tapaamista. Ongelma ilmeni tilanteessa, jossa opiskelija oli valinnut ajan, johon hän tapaamisen haluaisi, ja painanut kyseistä laatikkoa kalenterista. Tällöin opiskelijan varaama laatikko muuttui violetiksi (kuva 25). Käyttöliittymässä ei ole kuitenkaan mainintaa siitä mitä violetti väri tarkoittaa, jolloin opiskelijalle saattaa jäädä epäselväksi, että varaus onnistui ja tapaaminen on todellakin sovittu yhdellä ruudun painalluksella.

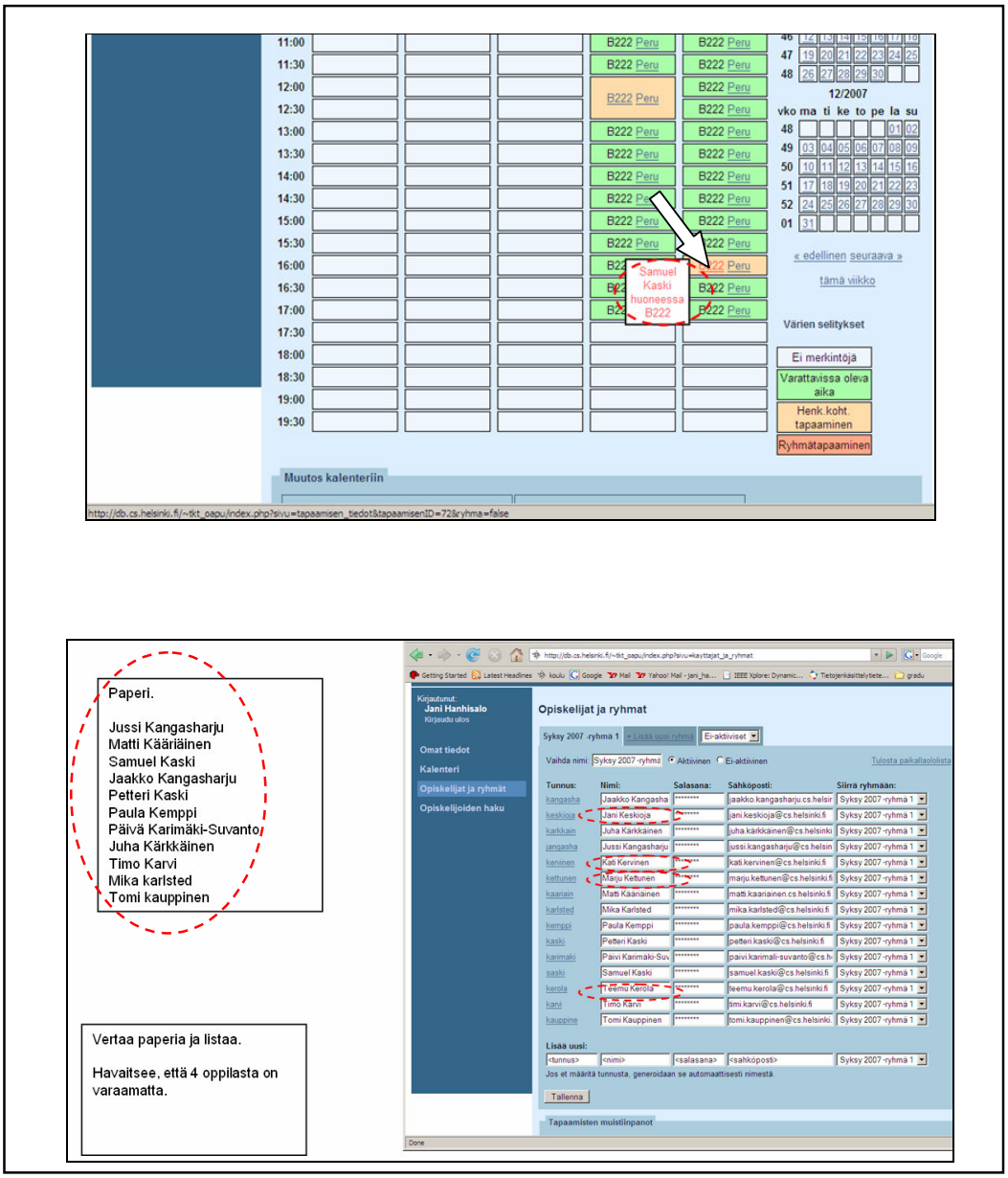


Kuva 25. Opittavuusongelmia Opeapuri-ryhmän kalenterinäkymässä.

6.4.5 Opettajatuutori selvittää varaustilanteen

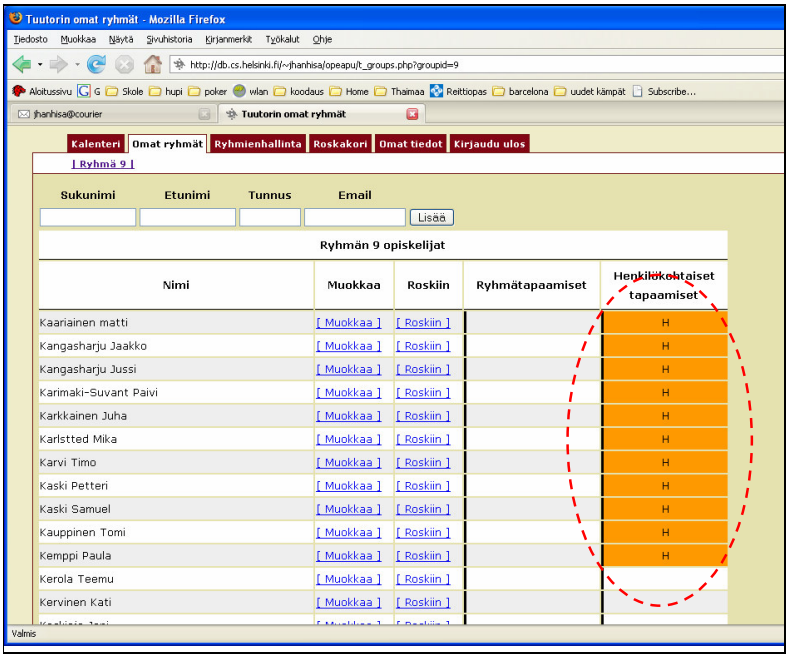
Henkilökohtaisille tapaamisille sovittujen aikojen lähestyessä opettajatuutorilla on tarve selvittää, ketkä ovat varanneet ajat ja mahdollisesti lähettää muistutukset henkilöille, jotka eivät ole muistaneet varata tapaamista. Opeapuri-ryhmän käyttöliittymästä puuttuu keskeinen toiminnallisuus, sillä varaustilanteen tarkistaminen oli hyvinkin vaivalloista, koska järjestelmässä ei ole sivua, josta kävisi kerralla selväksi, ketkä ovat tehneet varauksen ja ketkä eivät.

Kuvassa 26 on esitetty nykyinen keino selvittää, ketkä ovat varanneet tapaamisen. Viemällä hiiren kursorin varauksen päälle opettajatuutori näkee, kenelle opiskelijalle varaus on tehty. Opettajatuutorin on siis vietävä hiiri jokaisen tehdyn henkilökohtaisen tapaamisen päälle ja kirjoitettava opiskelijan nimi paperille muistiin. Näin opettajatuutori saa listan henkilöistä, jotka sillä hetkellä ovat sopineet tapaamisen. Tämän jälkeen opettajatuutori siirtyy Opiskelijat ja ryhmät -sivulle. Sieltä hän selvittää varaustilanteen etsimällä ryhmän opiskelijoiden listasta ne henkilöt, jotka eivät ole paperilistassa ja saa näin selville, ketkä eivät ole tehneet varausta. Huomioitavaa on myös, että opiskelijat voivat perua varauksia. Näin ollen joka kerta, kun opettajatuutori tarkistaa senhetkisen varaustilanteen, hän joutuu käymään läpi edellä kuvatun toimintoketjun.



Kuva 26. Varaajien selvittäminen Opeapuri-ryhmän käyttöliittymässä.

Kuvassa 27 on esitetty, miten Opeapu-ryhmä on ratkaissut samaisen ongelman sijoittamalla opiskelijalistaan sarakkeen, josta opettaja näkee, kuka on sopinut henkilökohtaisen tapaamisen.



Kuva 27. Varaustilanteen selvittämine Opeapu-ryhmän käyttöliittymässä.

Saadakseen selville, ketkä ovat varanneet henkilökohtaisen tapaamisen Opeapuri-ryhmän tapauksessa. Opettaja joutuu tekemään yhteensä 34 toimenpidevaihetta, kun vaiheiksi lasketaan hiiren vienti ruudun päälle, nimen kirjaaminen, siirtymät ja vertailut. Opeapu-ryhmän käyttöliittymässä sama vaatii kaksi toimenpidevaihetta, jotka ovat Omat ryhmät -sivulle siirtyminen ja tilanteen katsominen näytöltä. Sisään kirjautumista ei ole laskuissa mukana, koska se vaatii molemmissa tapauksissa saman määrän toimenpiteitä. Opeapuri-ryhmän puuttuvaa toiminnallisuutta voidaan siis pitää vakavana tehokkuusongelmana.

Seuraavaksi opettajatuutorilla on tarve lähettää sähköpostimuistutus niille opiskelijoille, jotka eivät vielä ole varanneet henkilökohtaista tapaamista. Simuloidussa tapauksessa heitä oli 4. Simuloitaessa sähköpostiosoitteiden selvittämistä Opeapu-ryhmän käyttöliittymästä havaittiin vakava tehokkuusongelma. Kuvasta 25 voi nähdä, että opiskelijoiden sähköpostiosoitteita ei ole listassa, vaan ne on mahdollista nähdä vasta opiskelijan tiedot esittävältä sivulta, jonne päästään painamalla Muokkaus-painiketta. Opettaja joutuu siis avaamaan ja sulkemaan muokkausikkunan jokaisen sellaisen opiskelijan kohdalla, jonka sähköpostiosoitteen hän haluaa selvittää, tässä tapauksessa

neljä kertaa. Järjestelmässä on siis selkeästi tehokkuusongelma sähköpostiosoitteiden selvityksessä, koska käyttäjä joutuu avaamaan neljä eri ikkunaa saadakseen sähköpostiosoitteet selville.

Seuraavassa vaiheessa opettaja tarkistaisi varaustilanteen vielä uudestaan lähempänä tapaamisia, joka on identtinen toiminto edelle esitetyn tarkastuksen kanssa. Täysin toinen käyttötapaus olisi selvittää, miten opettajatuutori toimii, kun tapaamiset on pidettyt ja joku opiskelijoista ei ole saapunut tapaamiseen tai ei ole varannut aikaa pyynnöistä huolimatta. Lisäksi simuloidun käyttötapauksen variaatiot, joissa ilmenisi esimerkiksi tarvetta vaihtaa sovittu tapaaminen, tai tilanne, jossa mikään opettajatuutorin ehdottamista ajoista ei sovi, olisivat tilanteita, joiden simulointi voisi paljastaa lisää ongelmakohtia. Tässä työssä ei kuitenkaan simulointia jatkettu tätä pidemmälle, koska jo ensimmäisten kolmen käyttötapauksen simulointi paljasti useita keskeisiä ongelmia ja puutteita ryhmien käyttöliittymissä, joita voidaan verrata vaatimusmäärittely-dokumentissa havaittuihin käyttöliittymäratkaisuja kiinnittäviin vaatimuksiin.

7. Vaatimusmäärittelydokumentin vaatimusten ja käyttötapausten yhteys käyttöliittymäongelmiin

Vaatimusmäärittelydokumenttien analysoinnin tuloksena saatiin selville, että lähes kaikki käyttötapaukset sitovat toiminnon ja sen toteutuksen käyttöliittymässä. Suurin osa käyttäjävaatimuksista ja järjestelmävaatimuksista sitoo ainoastaan sen, että järjestelmässä on oltava tietty toiminnallisuus. Simulointitestauksen tuloksena saatiin iso joukko käyttöliittymässä havaittuja ongelmia. Tässä luvussa tarkastellaan, löytyykö sitovien vaatimusten ja käyttöliittymässä havaittujen käytettävyysongelmien väliltä yhteyksiä vai onko ongelmien takana muita syitä.

7.1 Opiskelijoiden lisäämisen vaikeudet

Molempien ryhmien käyttöliittymissä yksi keskeisin tehokkuusongelma ja toisaalta puuttuva toiminnallisuus liittyi oppilaiden lisäämiseen (kuva 9 ja kuva 16). Molemmissa käyttöliittymissä opiskelijan luonti oli toteutettu ketjulla jossa opettaja täytä oppilaan tiedot ja paina Tallenna-painiketta. Tämä kuitenkin aiheutti käyttötapauksen läpiviennissä tehokkuusongelman, koska käyttäjä joutui painamaan jokaisen lisäyksen jälkeen Tallenna-painiketta.

Molempien ryhmien vaatimusmäärittelydokumentissa on sekä vaatimus oppilaan lisäämisen mahdollistamisesta että käyttötapaus, jossa on tarkka kuvaus siitä, miten lisäystoimintoa käytetään. Taulukossa 11 on Opeapuri-ryhmän käyttäjävaatimus ja käyttötapaus, jossa opiskelijan lisäys kuvataan. Käyttäjävaatimus ei sido käyttöliittymäratkaisua. Toiminnon se sitoo, mutta tässä tapauksessa se ei vielä ole haitallista. Käyttötapaus sen sijaan sitoo tarkasti, miten opiskelijan lisäys tulee järjestelmässä menemään. Kuvauksesta käy ilmi, että lisäys tullaan toteuttamaan ketjulla ”täytä tiedot ja paina Tallenna-painiketta”. Tässä on sidottu sekä toiminto että osa sen interaktiotavoista käyttöliittymässä. Voidaan todeta, että taulukossa 11 esitetty käyttötapauskuvaus on aiheuttanut toteutettuun käyttöliittymään selvän tehokkuus-ongelman.

Opiskelijoiden lisäys Opetuutori voi lisätä johonkin ryhmistään uusia opiskelijoita.

Käyttötapaus	**Opiskelijoiden lisääminen**
Käyttäjän rooli	Opetuutori
Käyttäjän toiminta	Avaa opiskelijanlisäyssivun. Syöttää tyhjiin kenttiin opiskelijan nimen (pakollinen tieto), käyttäjätunnuksen (pakollinen tieto, ohjelma voi myös generoida tunnuksen annetusta nimestä käyttäjän painaessa generointinappia), sähköpostiosoitteen, salasanan (voi jättää tyhjäksi) ja tarvittaessa vapaamuotoista tekstitietoa opiskelijasta, ja valitsee opiskelijalle ryhmän listasta (yhden omista ryhmistään.), ja painaa lisäysnappia.
Järjestelmän toiminta	Jos käyttäjä painaa generointinappia, järjestelmä generoi tunnusehdotuksen annetusta oppilaan nimestä. Tallentaa syötetyt tiedot tietokantaan ja luo uuden käyttäjän.
Poikkeustilanne	Pakollinen syötettävä tieto (nimi tai tunnus) puuttuu. Saman tunnuksen

Taulukko 11. Opiskelijoiden lisäämisen käyttäjävaatimus ja käyttötapaus Opeapuri-ryhmän vaatimusdokumentissa [Ekl07].

Havaittujen pienempien opittavuusongelmien, kuten kenttien liian pieni koko, ja vaatimusten väliltä ei ole löydettävissä yhteyttä. Ne ovat ilmeisesti syntyneet suunnittelu- tai toteutusvaiheessa, kun käyttöliittymä on toteutettu. Mielenkiintoinen havainto on myös se, että järjestelmä on toteutettu siten, että otsikot on sijoitettu kenttien sisään, vaikka käyttötapauksessa on nimenomaan maininta tietojen syöttämisestä tyhjiin kenttiin. Simuloinnissa otsikoiden sijoittelun kenttiin todettiin aiheuttavan turhaa mentaalista työtä.

7.2 Henkilökohtaisten tapaamisten varaustilanteen tarkistamisen vaikeus

Toinen keskeinen ongelma käyttöliittymässä on varaustilanteen tarkistamisen työläys. Ongelma esiintyy ainoastaan Opeapuri-ryhmän käyttöliittymässä. Vakavaksi tehokkuus-ongelmaksi simulointitestauksessa havaittiin kuvassa 26 esitetty tilanne, jossa opettajatuutori selvittää, ketkä oppilaista ovat tehneet varauksen ja ketkä eivät. Käyttöliittymää ei ole suunniteltu tukemaan tätä tehtävää laisinkaan.

Vaatimusmäärittelydokumentissa ei ole kirjattu minkäänlaista vaatimusta, jossa mainittaisiin, että opettajatuutorilla on tarve saada selville, ketkä eivät vielä ole varanneet henkilökohtaista tapaamista, joten sitä ei ole huomioitu käyttöliittymää suunniteltaessa. Ainoa asiaan vähän viittaava käyttötapaus on kalenterin tarkastelu, josta mainitaan vain, että opettajatuutorin on voitava silmäillä kalenteria.

Käyttötapauksessa ei määritellä tarkemmin, miksi tai missä tilanteissa opettajatuutori tarkastelisi kalenteria, joten sen käyttötarkoitus jää epäselväksi.

Vaatimusmäärittelydokumentissa ei ole vaatimusta, joka aiheuttaisi tämän ongelman. Ongelma johtuu ilmeisesti siitä, että vaatimusmäärittelyvaiheessa ei ole selvitetty käyttäjän todellisia työnkulkuja, vaan on vain listattu mieleen tulevia ja asiakkaalta saadut vaatimukset. Näin vaatimuksista on jäänyt pois työnkulun kannalta oleellinen vaatimus. Tämän seurauksena käyttöliittymästä puuttuu työnkulun vaatimia toimintoja, jotka mahdollistaisivat yksinkertaisen tavan tarkastella, ketkä opiskelija ovat sopineet ja ketkä eivät ole sopineet henkilökohtaisia tapaamisia.

7.3 30 minuutin varausjakso

Kolmas vakava käyttöliittymäongelma molempien ryhmien käyttöliittymissä on se, että ne tarjoavat opiskelijalle mahdollisuuden varata ainoastaan 30 minuutin aikajakson henkilökohtaiselle tapaamiselle. Haastattelun perusteella on selvästi havaittavissa, että osa opettajatuutoreista varaa henkilökohtaiselle tapaamiselle 60 minuuttia. Nyt tällaisen varauksen opettajatuutori voi tehdä vain Opeapuri-ryhmän käyttöliittymässä. Opeapuri-ryhmän käyttöliittymässä voi tehdä ainoastaan 30 minuutin pituisia varauksia.

Opeapuri-ryhmän vaatimusmäärittelydokumenttiin on kirjattu ei-toiminnallinen järjestelmävaatimus, jossa mainitaan, että ”Kalenteri sisältää ajat maanantaista perjantaihin klo 8-20. Päivät on jaettu tasaisesti 30 minuutin pituisiin osiin”. Vaatimus sitoo sen, että kalenterin tulee näyttää 30 minuutin aikalohkot, mutta se ei sido, että pitempiä lohkoja ei voisi varata. Käyttötapauksissa taas on, kuvattu miten opiskelija voi tapaamisen varata (taulukko 12). Käyttötapauksessa mainitaan, että ”valitsee kalenterista ajan”. Kuvauksesta saa sen käsityksen, että toiminto tosiaan etenee siten, että opiskelija valitsee kalenterista yhden ruudun ja painaa varaamispainiketta. Tällöin käyttötapaus ja järjestelmävaatimus yhdessä ovat osaltaan kiinnittäneet käyttöliittymäratkaisua ja aiheuttaneet kyseisen käytettävyysongelman.

Käyttötapaus	Henkilökohtaisen tapaamisen varaaminen (opetuutori)
Käyttäjän rooli	Opetuutori
Käyttäjän toiminta	Avaa kalenterisivun. Valitsee kalenterista ajan, valitsee opiskelijan ja painaa varaamisnappia.
Järjestelmän toiminta	Tallentaa tietokantaan henkilökohtaisen tapaamisen ja muuttaa tietokannassa ko. ajan varatuksi.
Poikkeustilanne	Aikaa ei ole valittu. Tietokantapalvelimeen ei saada yhteyttä.
Prioriteetti	1: Pakollinen

Taulukko 12. Henkilökohtaisen tapaamisen toimintoketjun sitova käyttötapaus Opeapuri-ryhmän vaatimusmäärittelydokumentissa [Ekl07].

Tärkein syy ongelmaan lienee kuitenkin puuttuva vaatimus, jossa mainittaisiin mahdollisuus muuttaa opiskelijoiden varaaman aikalohkon pituutta opettajatuutori-kohtaisesti. Tällöin ongelma poistuisi kokonaisuudessaan.

Opeapu-ryhmän käyttöliittymässä ilmeni sama käytettävyysongelma. Ryhmän järjestelmävaatimuksissa määritellään, että kalenteri jaetaan 30 minuutin aikajaksoihin. Näin ollen myös Opeapu-ryhmän järjestelmävaatimus on kiinnittänyt kalenterin rakenteen ja johtanut kyseiseen käytettävyysongelmaan.

7.4 Opettajatuutori ei voi varata henkilökohtaista tapaamista

Puuttuvaksi toiminnallisuudeksi Opeapu-ryhmän käyttöliittymässä havaittiin, että opettajatuutori ei voi varata ollenkaan henkilökohtaisia tapaamisia, vaan ainoastaan opiskelija voi. Opeapu-ryhmän vaatimusmäärittelydokumentissa on opettajatuutorille ainoastaan käyttäjävaatimus, jossa määritellään, että opettajatuutori voi lisätä itselleen sopivia yksilötapaamisaikoja. Dokumentissa ei kuitenkaan ole vaatimusta, jossa mainittaisiin, että opettajatuutori voi varata henkilökohtaisia tapaamisia.

Vaatimusmäärittelyvaiheessa ei ilmeisesti ole tullut esille tilannetta, jossa opettaja-tuutorin olisi tarvetta varata itse henkilökohtaisia tapaamisia. Näin vaatimusdokumenttiin ei ole tällaista vaatimusta kirjattu. Vaatimuksista siis puuttuu todellisen työnkulun vaatimaa toiminnallisuutta, jonka seurauksena myös käyttöliittymästä puuttuu kyseinen toiminnallisuus.

7.5 Turha navigointi kalenterissa

Opeapu-ryhmän käyttöliittymässä kalenterisivu on rakennettu siten, että kerralla näkyvissä on ainoastaan yksi viikko. Simuloitaessa todellista käyttötilannetta ratkaisu johti kalenteria käytettäessä turhaan navigointiin viikosta toiseen (kuva 23). Ei-toiminnallisissa järjestelmävaatimuksissa on määritelty, miten kalenterissa tulee näkyä

ainoastaan viikko kerralla (kuva 28). Lisäksi useista käyttötapauksista käy ilmi, että kalenterissa tulee olemaan kerralla näkyvissä ainoastaan viikko. Esimerkiksi taulukossa 13 oleva käyttötapaus kalenterin tarkastelusta olettaa myös, että kerralla näkyy kalenterissa kuluva viikko.

Kalenteri

Kalenteri sisältää ajat maanantaista perjantaihin klo 08-20. Päivät on jaettu tasaisesti 30 minuutin pituisiin osiin.

Kuva 28. Kalenterin rakenteen sitova järjestelmävaatimus Opeapuri-ryhmän vaatimusmäärittelydokumentissa [Ekl07].

Käyttötapaus	**Kalenterin tarkastelu**
Käyttäjän rooli	Opetuutori, Opiskelija
Käyttäjän toiminta	Avaa kalenterisivun. Tarkastelee kuluvan viikon ohjelmaa. Ohjelmassa näkyy tehdyt varaukset ja varattavissa olevat ajat tapaamisille. Näkymää voi vaihtaa tuleville viikoille.
Järjestelmän toiminta	Hakee tietokannasta kuluvan viikon aikataulun ja generoi siitä lukujärjestyksen näytettäväksi käyttäjälle.
Poikkeustilanne	Tietokantapalvelimeen ei saada yhteyttä.
Prioriteetti	1: Pakollinen

Taulukko 13. Kalenterin tarkastelu käyttötapaus Opeapuri-ryhmän käyttöliittymässä [Ekl07].

Vaatimusmäärittelyvaiheessa on selvästi sidottu kalenterin rakenne käyttöliittymässä. Ratkaisu aiheuttaa kuitenkin käyttöliittymään turhaa navigointia. Näin ollen voidaan todeta, että edellä kuvatut vaatimukset sitovat käyttöliittymäratkaisuja haitallisesti. Mielenkiintoinen havainto on, että Opeapu-ryhmän vaatimuksissa ei sidota kalenterin rakennetta eikä ryhmän käyttöliittymässä ole turhaa navigointia, vaan kalenterissa on noin kuukauden jakso kerralla näkyvissä.

7.6 Osa opiskelijan tiedoista erillisellä sivulla

Opeapu-ryhmän käyttöliittymässä havaittiin tehokkuusongelma selvitettäessä opiskelijoiden sähköpostiosoitteita muistutussähköpostin lähettämistä varten. Simuloinnissa opettajatuutori joutuu aluksi painamaan Muokkaa-painiketta, jotta hän saa opiskelijan sähköpostiosoitteen sisältävän lomakkeen avautumaan uuteen ikkunaan. Luettuaan tiedot opettaja sulkee sivun. Sama toimintoketju toistuu jokaisen oppilaan

kohdalla. Opettaja joutuu siis turhaan avaamaan ja sulkemaan yhden ikkunan jokaista sellaista oppilasta kohden, jolle hän on aikonut lähettää sähköpostia.

Vaatimusmäärittelydokumentissa on taulukossa 14 kuvattu käyttötapaus opiskelijan tietojen muokkaustavasta. Käyttötapauksessa mainitaan, että opiskelijan tiedot -painiketta painamalla päästään tilaan, jossa muokkaaminen on mahdollista. Tässä selvästi kiinnitetään se, että järjestelmässä tulee olemaan erillinen sivu, jolla tietoja voidaan muokata. Lisäksi käyttötapauskuvauksessa mainitaan, että opiskelijalistassa näkyy opiskelijan nimi. Tämä viittaa vahvasti siihen, että listassa ei tule olemaan muuta kuin nimi, ja muut tiedot sijaitsevat opiskelijan tiedot -painikkeesta aukeavassa ikkunassa. Käyttötapauksessa on lisäksi maininta, että käyttäjän on ennen opiskelijan tiedot -painikkeen painamista valittava opiskelija vieressä olevasta valintapainikkeesta. Tätä ei onneksi ole käyttöliittymään toteutettu, sillä se lisäisi toimintoketjuun vielä yhden vaiheen lisää.

Opiskelijan tietojen muuttaminen

Alkutila	Opettajatuutori haluaa muuttaa oman tuutoriryhmänsä opiskelijoiden henkilötietoja.
Lopputila	Opettajatuutori on saanut tehtyä haluamansa muutokset omaan tuutoriryhmäänsä ja on ”opiskelijan tiedot” -näkymässä.
Kuvaus	Opettajatuutori siirtyy siihen järjestelmän välilehteen, josta hän näkee omaan tuutoriryhmäänsä kuuluvien opiskelijoiden nimet. Opettajatuutori valitsee haluamansa opiskelijan tuutoriryhmästään painamalla opiskelijan vieressä olevaa valintanappia. Tämän jälkeen opettajatuutori painaa ”opiskelijan tiedot” -nappia, josta hän pääsee tilaan jossa opiskelijan henkilötietojen muokkaaminen on mahdollista. Opettajatuutori tekee tarvittavat muutokset opiskelijan tietoihin sekä kuittaa muutokset painamalla ”talleta muutokset” -nappia.
Poikkeukset ja rajoitukset	Opiskelijan käyttäjätunnusta ei voida muuttaa sellaiseksi joka on jo jollain järjestelmässä olevalla käyttäjällä.

Taulukko 14. Opeapu-ryhmän käyttötapaus opiskelijan tietojen muuttamisesta [Tuo07].

Voidaan todeta, että käyttötapaus, jossa kuvataan opiskelijan tietojen muokkaus, kiinnittää käyttöliittymäratkaisuja haitallisesti. Sen seurauksena myöhemmissä ohjelmistoprosessin vaiheissa toteutetussa käyttöliittymässä on edellä kuvattu tehokkuusongelma.

7.7 Käyttöliittymäongelmia, joiden syy ei löydy vaatimuksista

Havaituista käyttöliittymäongelmista useat ovat pieniä opittavuusongelmia kuten opiskelijan lisäyksessä kenttien liian pieni koko. Näihin ongelmiin ei vaatimusmäärittelystä löydy viitteitä. Voidaankin olettaa, että ne ovat syntyneet käyttöliittymän

toteutusvaiheessa. Tällöin niiden syntymisen syy lienee huolimattomuudessa ja kiireessä saada ohjelmisto valmiiksi, jolloin ei ole ollut aikaa keskittyä pieniin yksityiskohtiin.

Opeapu-ryhmän kalenterissa havaittiin opittavuusongelmaksi myös käyttöliittymäkomponenttien sijoittelu kalenterinäkymässä. Ongelmalle ei löydy syytä vaatimuksista. Ilmeisesti rakenne on suunniteltu vasta toteutusvaiheessa, eikä sen käytettävyyttä ole ehditty testaamaan.

Näyttäisi vahvasti siltä, että suuret käytettävyysongelmat, kuten vakavat tehokkuusongelmat ja puuttuva toiminnallisuus, ovat ongelmia, joiden syyt ovat vaatimusmäärittelyvaiheessa. Toisaalta taas opittavuusongelmat, jotka johtuvat esimerkiksi komponenttien sijoittelusta, ovat syntyneet vasta myöhemmissä prosessinvaiheissa, kun käyttöliittymän toteutus on aloitettu.

8. Tutkimuskysymysten analysointi

Opeapuri-ryhmän käyttötapauksista 12/31 liittyy simuloituun tavoitepohjaiseen käyttötapaukseen. Ryhmän toiminnallisista käyttäjävaatimuksista taas 11/28 liittyy simuloituihin käyttötilanteisiin. Siten tutkimus kattaa noin kolmasosan ryhmän kirjaamista vaatimuksista. Opeapu-ryhmän käyttötapauksista 11/42 liittyy simuloituihin käyttötapauksiin ja käyttäjävaatimuksista 9/31. Opeapu-ryhmän vaatimusten kokonaismääriä selittää osin se, että ryhmä oli kirjannut vaatimuksiin ja käyttötapauksiin toimintoja, jotka liittyivät ilmoitustauluun. Ryhmä on suunnitellut, että opettajat ja opiskelijat kommunikoivat toistensa kanssa ilmoitustaulun avulla. Ilmoitustaulua ei kuitenkaan koskaan toteutettu.

Vertaamalla vaatimuksia käyttöliittymäongelmiin havaittiin, että 3/15 eli 20 % Opeapuri-ryhmän käytettävyysongelmista johtui vaatimusmäärittelydokumenttiin kirjatuista vaatimuksista tai käyttötapauksista. Kaksi käyttöliittymäongelmaa johtuu käyttötapauksista ja yksi ei-toiminnallisesta järjestelmävaatimuksesta. Opeapu-ryhmällä sama lukema on 2/12 eli noin 17 %. Molemmissa käytettävyysongelmissa syy vaikuttaa vahvasti olevan käyttötapauksessa. Ongelmissa on mukana myös pienet opittavuusongelmat. Keskeinen havainto on, että vaatimuksista ja käyttötapauksista johtuvat ongelmat ovat kaikki hyvin vakavia ongelmia eli niiden seurauksesta käyttöliittymän käyttö on hyvinkin vaivalloista.

Seuraavissa aliluvuissa käsitellään tässä työssä saatuja vastauksia asetettuihin tutkimuskysymyksiin. Lisäksi esitellään myös muut työn tuloksista pääteltävät havainnot.

8.1 Vaatimusten vaikutukset käyttöliittymäsuunnitteluun

Ensimmäisenä tutkimuskysymyksenä on, sitovatko vaatimusmäärittelydokumenttiin kirjatut vaatimukset ja käyttötapaukset käyttöliittymää, ja mikäli sitovat, aiheuttavatko ne ongelmia käyttöliittymään. Tutkimuksen perusteella voidaan sanoa, että käyttötapaukset sitovat sekä toiminnon että sen interaktiotapoja käyttöliittymässä. Käyttäjävaatimukset ja järjestelmävaatimukset sitovat yleensä ainoastaan toiminnon, joka järjestelmän käyttöliittymän tulisi tarjota jollain tavalla.

Analysoitaessa simulointitestauksessa ilmenneitä käytettävyysongelmia havaittiin, että 2 Opeapuri-ryhmän vaatimusmäärittelydokumenttiin kirjaamista käyttötapauksista on sitonut selvästi käyttöliittymäratkaisua siten, että sen seurauksena myöhemmin toteutettuun käyttöliittymään on syntynyt käytettävyysongelmia. Ensimmäinen kiinnittävä käyttötapaus on luvussa 7.1 esitelty opiskelijoiden lisääminen opettajatuutorointi-ryhmään. Käyttötapauksessa kuvattu toteutustapa aiheuttaa käyttöliittymään vakavan tehokkuusongelman. Toinen haitallisesti kiinnittävä käyttötapaus on käyttötapaus, jossa on kuvattu, että kalenterissa on kerralla näkyvissä viikon kokoinen aikajakso. Simulointitestauksessa havaittiin, että ratkaisu aiheuttaa käyttöliittymässä turhaa navigointia.

Myös Opeapu-ryhmän vaatimuksissa on 2 haitallisesti käyttöliittymäratkaisuja sitovaa käyttötapausta. Ensimmäinen on siinäkin opiskelijoiden luonti ja toinen on opiskelijoiden tietojen muokkaus. Käyttötapauksessa kuvattiin, miten opiskelijoiden muut tiedot kuin nimi tulevat sijaitsemaan omassa ikkunassaan, joka aukeaa opiskelijalistasta Muokkaa-painiketta painamalla. Simuloinnissa havaittiin kuitenkin, että kyseinen ratkaisu johtaa siirtymiseen pääikkunan ja muokkausikkunan välillä.

Näyttäisi siltä, että mikäli vaatimusmäärittelydokumenttiin kirjataan käyttötapauksia, tullaan samalla sitoneeksi toimintojen toteutus käyttöliittymässä, ja näin saatetaan aiheuttaa jo vaatimusmäärittelyvaiheessa käyttöliittymään tehokkuusongelmia. Tosin vaikka lähes jokainen käyttötapaus kiinnittää käyttötapauksen toteutuksen käyttö-liittymässä, silti tutkimuksen tulosten perusteella ainoastaan hyvin pieni osa kiinnittävistä käyttötapauksista aiheuttaa vakavia tehokkuusongelmia.

Vaikuttaisi myös vahvasti siltä, että isoin riski käytettävyysongelmille on tilanteissa, joissa joudutaan ketjuttamaan useita eri käyttötapauksia tai toistamaan samaa

käyttötapausta useita kertoja peräjälkeen. Käyttötapauksessa kuvataan esimerkiksi vain yhden opiskelijan luonti, vaikka todellisessa käyttötilanteessa kaikki opiskelijat luodaan usein yhdellä kertaa. Käyttötapauksen kirjaaja ei kykene luontivaiheessa sitä kuitenkaan huomioimaan, koska hän vain miettii, miten yksi opiskelija voidaan luoda yksinkertaisimmin. Näin käyttöliittymästä jää helposti pois toistoja vaativien käyttötilanteiden tukeminen.

Vakavimmat käyttöliittymässä havaitut ongelmat ovat hieman yllättäen työnkulun kannalta oleellisten toimintojen puuttumisen aiheuttamat ongelmat. Niiden keskeisin syy on toiminnon määrittävien vaatimusten puuttuminen vaatimusmäärittelydokumentista. Vaikuttaa vahvasti siltä, että vaatimusmäärittelyvaiheessa ei ole kyetty hahmottamaan riittävän hyvin käyttäjän todellisia työnkulkuja, minkä seurauksena on vaatimuksista jäänyt keskeisiä toimintoja pois.

Käyttäjä- ja järjestelmävaatimukset kiinnittivät toiminnallisuuden, mutta eivät sen interaktiotapoja käyttöliittymässä. Tutkimuksessa ei havaittu, että dokumenttiin kirjatut vaatimukset kiinnittäisivät toimintoja siten, että niistä on haittaa käyttöliittymälle. Tutkimuksessa havaittiin kyllä, että vaatimuslistoista puuttuu työnkulkujen kannalta keskeisiä toimintoja. Tutkimuksen aikana heräsi lisäksi kysymys, ovatko kaikki vaatimuslistoihin kirjatut vaatimukset todella tarpeellisia. Kysymykseen ei voida tämän työn pohjalta antaa vastausta, koska se olisi vaatinut kaikkien keskeisten käyttötapausten selvittämisen. Turhiin vaatimuksiin viittaa se, että vaikka suoritettu tavoitepohjainen käyttötapaus kattaa kokonaisuudessaan keskeisimmät tapaamisten sopimiseen liittyvät työnkulut, niin silti se kattaa vain noin kolmasosan Opeapuri-ryhmän kirjaamista vaatimuksista ja käyttötapauksista. Opeapu-ryhmän käyttäjävaatimuksista simulointi kattaa kolmasosan ja käyttötapauksista vain neljäsosan. Näiden lukujen pohjalta näyttäisi siltä, että vaatimuksissa saattaa olla turhia toimintoja, joille ei löydy mitään järkevää käyttötilannetta..

8.2 Käyttötilanteiden puuttumisen vaikutukset

Toisena tutkimuskysymyksenä oli selvittää, aiheuttaako todellisten käyttötilanteiden puuttuminen vaatimusmäärittelyvaiheessa käyttöliittymään selviä tehokkuusongelmia ja jääkö käyttöliittymästä puuttumaan käyttäjän käyttötilanteiden kannalta tarpeellista toiminnallisuutta tai tietosisältöä, koska mukana ei ole aitoja käyttäjiä. Tutkimuksessa havaittiin useita käyttötilanteita, joissa käyttöliittymästä puuttui käyttötilanteen tehokkaaseen toteuttamisen tarvittavaa toiminnallisuutta. Puuttuvaa toiminnallisuutta:

- opiskelijoiden massasyöttö (Opeapuri ja Opeapu),
- henkilökohtaisten tapaamisten varaustilanteen selvittäminen helposti (Opeapuri),
- opettajatuutorin olisi voitava varata henkilökohtaisia tapaamisia (Opeapu) ja
- tunnusten ja salasanojen lähetys opiskelijoille kootusti (Opeapuri ja Opeapu).

Lisäksi mielenkiintoinen havainto on, että Opeapu-ryhmä vaatimuksissa ja käyttötapauksissa ei ole laisinkaan mainintaa ryhmän luonnista. Käyttöliittymässä on kuitenkin ryhmien luonti toteutettu. Vaatimuksista on puuttunut oleellinen toiminnallisuus, mutta sitä ei ole havaittu vaatimusmäärittelydokumenttia validoitaessa. Ryhmä on kuitenkin ilmeisesti havainnut prosessin myöhemmissä vaiheissa, että järjestelmässä on syytä olla ryhmän luonnin mahdollistava toiminnallisuus.

Simulointitestauksessa havaittujen puutteiden pohjalta näyttää vahvasti siltä, että todellisten käyttötilanteiden puute on johtanut keskeisen toiminnallisuuden puuttumiseen. Ryhmät eivät ehkä ole kyenneet itse hahmottamaan, miten ohjelmistoa tullaan todellisuudessa käyttämään, ja sen seurauksena vaatimuksista on jäänyt oleellisia toimintoja pois. Lisäksi huomioitavaa on, että puuttuvaa toiminnallisuutta löytyy molempien ryhmien käyttöliittymistä. Voidaan siis epäillä, että vikaa voi olla myös menetelmässä, eikä ainoastaan esimerkiksi opiskelijoiden henkilökohtaisissa taidoissa.

8.3 Kehitysideoita

Tämän työn pohjalta voidaan sanoa, ettei ohjelmistotuotantoprojekteissa käytetty prosessimalli tuota ainakaan parasta mahdollista käyttöliittymää. Keskeisimmät puutteet vaikuttavat tämän työn perusteella olevan todellisten käyttötilanteiden selvittämisen puuttuminen ja käyttöliittymäsuunnittelun myöhäinen toteutusvaihe.

Näistä vakavin puute on se, että vaatimusmäärittely toteutetaan usein pelkästään ryhmän sisäisesti. Mikäli haastatteluja tehdään, haastateltavana on usein asiakas. Tällaisen menettelyn seurauksena vaatimusmäärittelydokumenttiin kirjataan listoja yksittäisistä toiminnoista ilman varsinaista tietämystä, onko toiminnolle todellista tarvetta ja missä tilanteissa toimintoa olisi tarkoitus käyttää. Parempi vaihtoehto olisi selvittää vaatimusmäärittelyn alussa järjestelmän keskeisimmät käyttötilanteet haastattelemalla järjestelmän tulevia käyttäjiä. Menetelmän tulisi olla sellainen, että se ohjaisi selvittämään käyttötilanteita ja käyttäjän tavoitteita yksittäisten toimintojen sijaan. Tällaisia menetelmiä olisivat esimerkiksi käyttäjätarkkailu tai kontekstuaalinen haastattelu. Haastattelun avulla saataisiin kuva siitä, missä tilanteissa ohjelmaa käytetään. Lisäksi käyttäjän työnkulkujen avulla olisi helpompi rajata ohjelmisto vain

keskeisimpiin toimintoihin, koska usein projekteissa vaatimuksiin kirjataan paljon toimintoja, joita ei tulla koskaan toteuttamaan, kuten esimerkiksi Opeapu-ryhmän ilmoitustaulu, johon oppilaat voisivat kirjoitella viestejä toisilleen.

Toinen keskeinen työssä havaittu ongelma on, että vaatimusmäärittelydokumenttiin kirjatut käyttötapaukset kiinnittävät myöhemmin toteutettavia käyttöliittymäratkaisuja. Osa kiinnittää ratkaisuja niin, että niistä aiheutuu käyttöliittymään ongelmia. Ongelma voitaisiin mahdollisesti poistaa suunnittelemalla käyttöliittymä GUIDe-mallin mukaisesti jo ennen vaatimusmäärittelyvaihetta suoraan tavoitepohjaisista käyttötapauksista. Näin varmistettaisiin myös, että aiemmin selvitettyjen käyttötilanteiden suoritus olisi mahdollista. Näin vaatimusmäärittelydokumenttiin ei olisi tarvetta kirjata laisinkaan käyttötapauksia, vaan sen tilalle voitaisiin dokumentoida tavoitepohjaiset käyttötapaukset ja suunnitellun käyttöliittymän näyttökuvat. Lisätutkimusta on kuitenkin tehtävä, että voitaisiin varmistaa, voitaisiinko myös toimintolistat korvata näyttökuvilla.

Jatkotutkimuksissa olisi mielenkiintoista selvittää, miten opiskelijat kykenisivät toteuttamaan käyttäjähaastattelut ja luomaan niiden pohjalta käyttöliittymäsuunnitelman. Toinen mielenkiintoinen kysymys olisi, miten ohjelmistotuotantoprojekti-kurssin ryhmien toteuttamat käyttöliittymät eroaisivat, mikäli sama projekti toteutettaisiin käyttäen perinteistä vesiputousmallia ja GUIDen mukaista prosessimallia. Lisäksi olisi lisäksi mielenkiintoista selvittää, kuinka moni vaatimuksista ja käyttötapauksista on todellisten käyttötilanteiden kannalta täysin turhia, koska tässä työssä sitä ei pystytty selvittämään.

9. Yhteenveto

Tässä työssä tutkittiin kahden Helsingin yliopiston ohjelmistotuotanto-kurssilla toteutetun opiskelijaprojektin vaatimusmäärittelyvaiheen vaikutuksia myöhemmin toteutettavaan käyttöliittymäsuunnitteluun. Työn ensimmäisessä vaiheessa kirjattiin ylös vaatimusmäärittelydokumenteista jollain tavalla käyttöliittymäratkaisuja sitovat vaatimukset. Toisessa vaiheessa luotiin tavoitepohjainen käyttötapaus pohjautuen todellisen käyttäjälle tehtyyn kontekstuaaliseen haastatteluun. Tämän jälkeen molemmille käyttöliittymille tehtiin simulointitestaus, jossa etsittiin paras polku käyttötapauksen läpiviemiselle. Sen tuloksena saatiin joukko käyttöliittymäongelmia. Viimeisessä vaiheessa etsittiin käyttöliittymäongelmille syitä vaatimusmäärittely-dokumentteihin kirjatuista käyttötapauksista ja muista vaatimuksista.

Tämän työn keskeisimpänä tuloksena havaittiin, että tutkituissa opiskelijaprojekteissa käyttötapaukset kiinnittävät käyttöliittymän toiminnallisuuden ja sen toteutuksen käyttöliittymässä yksityiskohtaisesti. Osa käyttötapauksien kiinnittämistä käyttöliittymä-ratkaisuista on sellaisia, että ne aiheuttavat käyttötilanteen suorituksessa käytettävyysongelmia. Lisäksi havaittiin, että käyttäjävaatimukset ja järjestelmä-vaatimukset sitovat pääsääntöisesti ainoastaan toiminnon, mutta eivät sen interaktiotapoja käyttöliittymässä. Työssä ei havaittu niiden aiheuttavan käyttöliittymään ongelmia. Vaikutti kuitenkin siltä, että vaatimuslistoissa on useita toimintoja, joiden tarpeellisuus on hyvinkin kyseenalainen.

Toinen keskeinen tulos on se, että kummassakaan käyttöliittymässä ei ole kaikkia käyttötilanteiden tehokkaan toteuttamiseen vaadittavia toiminnallisuuksia. Tässä työssä tehdyn tutkimuksen pohjalta näyttää vahvasti siltä, että ryhmien vaatimusmäärittely-vaiheessa ei ole ollut todellisia käyttäjiä mukana, ja sen seurauksena todellisten käyttötilanteiden kannalta keskeisiä toiminnallisuuksia ei ole kirjattu vaatimusmäärittely-dokumenttiin.

Yhteenvetona keskeisin havainto on, että perinteisellä tavalla toteutettu vaatimusmäärittely ilman todellisen käyttäjän työnkulkujen selvittämistä näyttäisi johtavan käyttöliittymäratkaisuihin, joissa toiminnallisuus ei vastaa todellisia työnkulkuja. Lisäksi vaikuttaa vahvasti siltä, että käyttötapaukset sitovat hyvin vahvasti käyttöliittymäratkaisuja ja aiheuttavat käyttöliittymään vakavia tehokkuusongelmia.

Jatkotutkimuksessa olisi syytä selvittää, millä tavoin tietojenkäsittelytieteen laitoksen ohjelmistotuotantoprojekteja voitaisiin viedä siihen suuntaan, että käyttäjien todelliset työnkulut olisi helpompi selvittää. Toiseksi olisi tutkittava keinoja, joiden avulla käyttöliittymäsuunnitelmakin saataisiin toteutettua mahdollisimman aikaisessa vaiheessa, jotta vaatimusmäärittelyvaiheessa tehdyt ratkaisut eivät kiinnittäisi käyttöliittymäratkaisuja haitallisesti. Tämä työ on kuitenkin osoittanut, että perinteinen vesiputousmallin mukainen vaatimusmäärittelyn avulla käyttöliittymään syntyy käytettävyysongelmia ja käyttötilanteiden tehokkaaseen suorittamiseen vaadittuja toimintoja jää käyttöliittymästä uupumaan.

Lähteet

Bia91 Bias, R., **Interface-Walkthroughs: efficient collaborative testing**. IEEE Software, Volume 8, No. 5, 1991, sivut 94-95.

BeH98 Beyer, H. ja Holtzblatt, K., **Contextual design: Defining customer-centered systems**. Morgan Kaufmann Publishers Inc., San Francisco, USA, 1998.

BeH99 Beyer, H. ja Holtzblatt, K., **Contextual Design**. Interactions, Volume 6, No. 1, 1999, sivut 32-42.

Boe06 Boehm, B., **A View of 21th and 21st Century Software Engineering**. *Proc. 28th Annual International Conf. on Software engineering*, Shangai, China, 2006, sivut 12-29.

Bra02 Bray, I. K, **An Introduction to Requirements Engineering**. Addison-Wesley, Essex, England, 2002.

Ekl07 Eklund, K. et al. **Opeapuri - Vaatimusdokumennti.** Ohjelmistotuotantoprojekti -kurssin vaatimusmäärittelydokumentti, Helsingin yliopisto, tietojenkäsittelytieteen laitos, 2007. http://www.cs.helsinki.fi/group/opeapuri/vaatimusmaarittely/vaatimusmaarittely_v1.pdf [tarkistettu 20.10.2007]

EnR03 Endres, A. ja Rombach, D., **A Handbook of software and systems engineering: Empirical observations, laws and theories**. Addison-Wesley, Essex, England, 2003.

GuS05 Guimarães, L. R. ja Souza Vilela, P. R., **Comparing software development models using CDM**. *Proc. 6th Annual International ACM SIGITE Conf, Information Technology Education* , Newark, NJ, USA, Lokakuussa 2005, sivut 339–347.

Han07a Hanhisalo, J., **Opeapuri-ryhmän käyttöliittymän simulointitestauksen tuloksena syntynyt kuvasarja**. Helsingin yliopisto, 2007. http://www.cs.helsinki.fi/u/jhanhisa/gradu/simulointitestaus_opeapuri_2007.pdf [tarkistettu 25.10.2007]

Han07b Hanhisalo, J., **Opeapu-ryhmän käyttöliittymän simulointitestauksen tuloksena syntynyt kuvasarja**. Helsingin yliopisto, 2007. http://www.cs.helsinki.fi/u/jhanhisa/gradu/simulointitestaus_opeapu_2007.pdf [tarkistettu 25.10.2007]

HaR98 Hackos, J. T. ja Redish, J. C., **User and task analysis for interface design**. John Wiley & Sons, New York, USA, 1998.

Ket07 Kettunen, M., **Vaatimusten ja käyttöliittymänratkaisujen suhde nykyisessä vaatimusmäärittelyssä ja kuinka prosessia voisi kehittää.** Seminaariesitys, Helsingin yliopisto, tietojenkäsittelytieteen laitos, 2007.

LeW97 Lewis C. ja Wharton C., **Cognitive walkthrough**. Teoksessa Helander M., Landauer T., Pradhu P., Handbook of human-computer interaction. Elsevier Science, 1997, s. 717-732.

Laa04 Laakso, S. A. ja Laakso, K-P., **Hyvän käyttöliittymän varmistaminen GUIDe-prosessimallilla**. Julkaisematon artikkeli, Helsingin yliopisto, tietojenkäsittelytieteen laitos, 2004. http://www.cs.helsinki.fi/u/salaakso/papers/GUIDe-suomeksi.pdf [tarkistettu 19.09.2007]

Laa06a Laakso, S. A., **Kenttätutkimukset**. Käyttöliittymät II -kurssin luentomoniste, Helsingin yliopisto, tietojenkäsittelytieteen laitos, 2006. http://www.cs.helsinki.fi/u/salaakso/kl2-2006/Kayttoliittymat2-Luento3-Kenttatutkimukset.pdf [tarkistettu 19.09.2007]

Laa06b Laakso, S. A., **Tavoitepohjaiset käyttötapaukset**.
Käyttöliittymät II -kurssin luentomoniste, Helsingin yliopisto, tietojenkäsittelytieteen laitos, 2006.
http://www.cs.helsinki.fi/u/salaakso/kl2-2006/Kayttoliittymat2-Kayttotapausten-tarkistuslista-Sari-A-Laakso.pdf [tarkistettu 27.09.2007]

Laa06c Laakso, S. A., **Käyttöliittymien arviointimenetelmät**.
Käyttöliittymät II -kurssin luentomoniste, Helsingin yliopisto, tietojenkäsittelytieteen laitos, 2006.
http://www.cs.helsinki.fi/u/salaakso/kl2-2006/Kayttoliittymat2-Luento7-Simulointitestaus-hyodyllisyyslapik-ja-muut.pdf [tarkistettu 02.10.2007]

Laa06d Laakso, S. A. ja Latva-Koivisto A., **Käyttöliittymät opetusmoniste**.
Käyttöliittymät I -kurssin luentomoniste, Helsingin yliopisto, tietojenkäsittelytieteen laitos, sarja D-2006-1, 2006.
http://www.cs.helsinki.fi/u/salaakso/papers/Kayttoliittymat-opetusmoniste-2006.pdf [tarkistettu 05.10.2007]

Lau02 Lauesen, S., **Software requirements: styles and techniques**.
Addison-Wesley, London, UK, 2002.

Lau05 Lauesen, S., **User interface design: A software engineering perspective**.
Pearson/Addison-Wesley, Essex, England, 2005.

Nel03 Neil, C. J. ja Laplante P. A., **Requirments Engineering: The State of the Practice**.
IEEE Software, Volume 20, No. 6, 2003, s 40-45.

NiM90 Nielsen J. ja Molich R., **Heuristic Evaluation of User Interfaces**.
CHI'90 Conf. Proc., ACM, New York, 1990, s. 249-256.

Nie93 Nielsen J., **Usability Engineering**.
Academic Press, San Diego, CA, 1993.

Pae03 Paetsch, F. ja Eberlein, A. ja Maurer, F., **Requirements Engineering and Agile Software Development**. *Proc 12th IEEE International Workshops WET ICE Conf, Enabling Technologies: Infrastructure for Colloborative Enterprises*, Kesäkuussa 2003, sivut 208-313.

PaK03 Paech, B. ja Kohler, K., **Usability Engineering Integrated with Requirments Engineering**. *Proc. of international Conf. on Software Engineering*, USA, 2003, sivut 36-40. [http://www.se-hci.org/bridging/icse/Final.pdf, 4.9.2007].

Pep00 Peters, J. F. ja Pedrycz, W., **Software Engineering: An Engineering Approach**. John Wiley & Sons, New York, USA, 1998.

Pfl98 Pfleeger S., **Software engineering: theory and practice**. Prentice-Hall, Inc., NJ, USA, 1998.

Pre97 Pressman, R. S., **Software engineering: A practitioner's approach**. McGraw-Hill, USA, 1997.

Rob03 Robinson, W. N., Pawlowski, S. D. ja Volkov V., **Requirements Interaction Management**. ACM Computing Surveys (CSUR) archive, Volume 35, No. 2, kesäkuu 2003, s.132-190.

Roy70 Royce, W. W., **Managing the Development of Large Software Systems**. *IEEE Western Conference*, 1970, sivut 1-9.

Som01 Sommerville Ian, **Software Engineering (6th edition)**. Addison-Wesley, Essex, England, 2001.

Tai06 Taina, J., **Female-only Software Engineering Student Teams- a Case Study**. *Proc. 9th International Conference on Engineering Education, San Juan, Puerto Rico, USA, 2006.*

Tai07 Taina, J., **Kurssimateriaali**.
Ohjelmistotuotanto -kurssin luentomoniste. Helsingin yliopisto, tietojenkäsittelytieteen laitos, 2007.
http://www.cs.helsinki.fi/u/taina/ohtu/k-2007/pdf/Ohjelmistotuotanto6d.pdf [tarkistettu 27.9.2007].

Tuo07 Tuominen, P. et. al., **Opeapu - vaatimusdokumentti**.
Ohjelmistotuotantoprojekti -kurssin vaatimusmäärittelydokumentti, Helsingin yliopisto, tietojenkäsittelytieteen laitos, 2007.

WiR96 Wixon, D. R. ja Ramey J., **Field methods casebook for software design**.
John Wiley & Sons, New York, USA, 1996.

Wix02 Wixon, D. R. et. al., **Usability in Practice: Field Methods Evolution and Revolution**. *Conf. on Human Factors in Computing Systems*, ACM CHI '02, NY, USA, 2002, sivut 880-884.

Liite 1. Opeapuri- ja Opeapu-projektin aihekuvaus

Tietojenkäsittelytieteen laitoksen uudet opiskelijat osallistuvat Opettajatuutorointi-kurssille, jossa heidän opintojensa edistymistä seurataan kandin tutkintoon asti. Tuutorointiin kuuluu lukukausittain pidettävä pakollinen henkilökohtainen tapaaminen ja samoin lukukausittain pidettävä pakollinen tai vapaaehtoinen ryhmätapaaminen.

Opettajatuutorit tarvitsevat avuksi työkalun, jonka avulla he voivat hallita henkilökohtaisia tapaamisia ja ryhmätapaamisia. Työkalussa tulee ainakin olla kalenteri, jonne opettajatuutor voi merkitä hänelle sopivia henkilökohtaisten tapaamisten aikoja, joista tuutoroivat voivat sitten varata haluamansa ajan. Lisäksi opettajatuutor tarvitsee keinot kirjata vapaamuotoisesti ylös tuutoroinnissa käsitellyt asiat. Ryhmätapaamisis varten opettajatuutorointi tarvitsee nimilistan ja keinon kirjata läsnäolot ylös tapaamisen jälkeen.

Tuutoroitavat voivat varata aikoja henkilökohtaiselle tapaamiselle ja tarkistaa heistä kirjatut asiat.

Sekä tuutorit että tuutoroitavat kirjautuvat järjestelmään järjestelmän sisäisellä käyttäjätunnuksella ja salasanalla. Järjestelmän superkäyttäjä perustaa käyttäjätunnukset ja salasanat järjestelmässä.

Toteutusympäristö:

Ohjelmisto toteutetaan WWW-selaimelle ryhmän haluamalla kielellä. Järjestelmän alla oleva tietokanta on laitoksella käytettävä Oraclen tietokanta.